



MSX FAN SERIES 1

マシン語入門(基礎編)

MIA

MSX FAN SERIES



入門(基礎編)



MSX FAN SERIES

入門基礎編

まえがき

本書は、これからMSXを使ってマシン語を学ぼうとする人のための入門書で、特にマシン語でプログラムをつくるための基礎的なことについて書かれています。

本書の主な内容は、コンピュータの基本構成、2進数・10進数・16進数など数の表現法と2進演算などの一般知識、MSXに使われているCPU（Z80A）の内部構成と機能の紹介、マシン語命令の説明などです。また、プログラムをつくるためのツールであるモニタ、アセンブラについても説明しています。なお、MSX用のモニタ・アセンブラのリストを第4章で公開しましたので、これを入力すれば本格的なマシン語プログラムをつくることができます。最後には、簡単ですがマシン語プログラムの作成方法やフローチャートの説明もしています。

また、巻末には付録としてZ80ACPUの命令表、マシン語・ニモニック対応表などを掲載しました。

本書がマシン語学習、プログラムづくりにいささかなりともお役に立てば幸いです。

1983年 12月 大貫広幸

■ まえがき

■ 第1章 マシン語のための予備知識 — 1

- 1. マシン語の長所と短所 — 2
- 2. マシン語の表わしかた — 5

■ 第2章 基礎知識 — 11

1. コンピュータの構造 — 12

- ①ハードウェアとソフトウェア — 12
- ②コンピュータの構成 — 12
- ③CPU — 13
- ④メモリ — 14
- ⑤入出力装置 — 15
- ⑥バス — 16

2. マイコンで扱う数 — 18

- ①数の表現法 — 18
- ②2進数と16進数 — 19
 - ・ 2進数→16進数変換 — 20
 - ・ 16進数→2進数変換 — 20
 - ・ 16進数→10進数変換 — 21
 - ・ 10進数→16進数変換 — 21
- ③負数と小数点以下の表現方法 — 21
 - 1) 負数の表現 — 21
 - 2) 小数点以下の数の表現 — 23
- ④その他の表現法 — 23
 - 1) BCD — 23
 - 2) 浮動小数点 — 24

3. 2進演算 — 25

- ①算術演算 — 25
 - 1) 加算 — 25
 - 2) 減算 — 27
 - 3) 2の補数表示と加減算 — 28
- ②論理演算 — 29
- ③シフトとローテイト — 31
 - 1) シフト — 31
 - 2) ローテイト — 32

■ 第3章 Z80Aのマシン語命令 ————— 33

1. Z80ACPU について ————— 34

①Z80Aのレジスタ構成	35
1) アキュムレータ(A, A')とフラグ・レジスタ(F, F')	35
2) 汎用レジスタ(B, C, D, E, H, L, B', C', D', E', H', L')	36
3) プログラム・カウンタ(PC)	36
4) スタック・ポインタ(SP)	37
5) インデックス・レジスタ(IX, IY)	37
6) インタラプト・ベクトル・レジスタ(I)	37
7) メモリ・リフレッシュ・レジスタ(R)	37
②割込みについて	38

2. Z80Aのニモニックとマシン語 ————— 39

①Z80Aのアドレッシング・モード	40
②データの転送, 交換	41
1) 8ビット・ロード命令	42
2) 16ビット・ロード命令	44
3) PUSH命令, POP命令	44
4) データの交換命令	46
③ブロック転送, ブロック・サーチ	47
1) ブロック転送命令	47
2) ブロック・サーチ命令	49
④演算命令	51
1) 8ビット演算命令	51
2) 16ビット演算命令	56
⑤ローテイト, シフト	58
1) ローテイト命令(RLD, RRD命令を除く)	59
2) ローテイト・ディジット命令(RLD, RRD命令)	59
3) シフト命令	60
⑥ビット操作, フラグ操作	60
1) ビット操作命令	60
2) フラグ操作命令	62
⑦ジャンプ, コール, リターン命令	62
1) ジャンプ命令	62
2) コール, リターン命令	64
3) リスタート命令	67
⑧入出力命令	67
1) IN, OUT命令	68
2) ブロック入出力命令	68
⑨CPUコントロール	69

■ 第4章 モニタ・アセンブラ ————— 71

1. モニタ・アセンブラの概要 ————— 72

2. 起動方法 ————— 73

① アセンブラの起動 ————— 74

② モニタの起動 ————— 74

3. スtring・サーチの使用法 ————— 75

4. マシン語モニタの使用法 ————— 75

1) メモリ・ダンプ(Dコマンド, DSコマンド) ————— 76

2) メモリの内容表示, 変更(Sコマンド) ————— 77

3) レジスタの内容表示, 変更(Xコマンド) ————— 77

4) マシン語プログラムの実行(Gコマンド) ————— 78

5) オブジェクトのロード(Rコマンド) ————— 78

6) BASICへ戻る(Bコマンド) ————— 79

5. セルフ・アセンブラの説明 ————— 80

① ソース・プログラムの形式 ————— 80

1) 文の構成 ————— 80

2) 文字 ————— 81

3) ラベル ————— 81

4) ニモニク ————— 82

5) オペランド ————— 83

6) コメント ————— 84

② プログラム・リストとラベル・リスト ————— 84

③ エラーメッセージ ————— 86

6. モニタ・アセンブラの入力方法 ————— 87

■ 第5章 マシン語プログラムの作成方法 113

1. プログラムの作成方法 ————— 114

2. フローチャート(流れ図) ————— 116

3. マシン語プログラムの作成例 ————— 117

■ 参考文献 ————— 124

■ Appendix(付録) ————— 125

■ 索引 ————— 151



1章 マシン語のための基礎知識

ここでは、マシン語とはどういうものなのか、どう表わされるのか、BASICとはどう違うのかなど、マシン語を学ぶ前に知っておくべき予備知識を解説しましょう。

マシン語の長所と短所

BASICとマシン語で同じ働きをするプログラムをつくり、実際に実行させてマシン語プログラムの長所と短所を考えてみましょう。プログラムは4つのスプライト⁽¹⁾を上下左右に動か

すだけの簡単なものです。

リスト1-1は**MSX-BASIC**⁽²⁾によるプログラムで、リスト1-2がマシン語によるプログラムです。リスト1-1と1-2は、どち

リスト 1-1 MSX-BASICによるサンプルプログラム

```

10
20 Sample Program No. 1
30 (MSX BASIC)
40
100 SCREEN 2,2
110 A$=""
120 FOR I=1 TO 32:READ B$:A$=A$+CHR$(VAL("&H"+B$)):NEXT:SPRITE$(0)=A$
130 DATA 01,03,07,0F,1F,3E,7C,F8,F8,7C,3E,1F,0F,07,03,01
140 DATA 80,C0,E0,F0,F8,7C,3E,1F,1F,3E,7C,F8,F0,E0,C0,80
150 A$=""
160 FOR I=1 TO 32:READ B$:A$=A$+CHR$(VAL("&H"+B$)):NEXT:SPRITE$(1)=A$
170 DATA 01,02,04,08,18,3C,7E,FF,FF,7E,3C,18,08,04,02,01
180 DATA 80,40,20,10,18,3C,7E,FF,FF,7E,3C,18,10,20,40,80
190 A$=""
200 FOR I=1 TO 32:READ B$:A$=A$+CHR$(VAL("&H"+B$)):NEXT:SPRITE$(2)=A$
210 DATA 01,02,04,08,11,23,47,8F,8F,47,23,11,08,04,02,01
220 DATA 80,40,20,10,88,C4,E2,F1,F1,E2,C4,88,10,20,40,80
230 A$=""
240 FOR I=1 TO 32:READ B$:A$=A$+CHR$(VAL("&H"+B$)):NEXT:SPRITE$(3)=A$
250 DATA 01,03,07,0F,1F,27,43,81,81,43,27,1F,0F,07,03,01
260 DATA 80,C0,E0,F0,F8,E4,C2,81,81,C2,E4,F8,F0,E0,C0,80
500
510 X0=50:X1=200:X2=78:X3=162
520 Y0=46:Y1=130:Y2=13:Y3=155
530 D0=+1:D1=-1:D2=+1:D3=-1
540 FOR I=1 TO 10000
550 PUT SPRITE 0,(X0,Y0),1,0
560 X0=X0+D0
570 IF X0<28 THEN D0=+1
580 IF X0>212 THEN D0=-1
590 PUT SPRITE 1,(X1,Y1),15,1
600 X1=X1+D1
610 IF X1<28 THEN D1=+1
620 IF X1>212 THEN D1=-1
630 PUT SPRITE 2,(X2,Y2),7,2
640 Y2=Y2+D2
650 IF Y2<1 THEN D2=+1
660 IF Y2>174 THEN D2=-1
670 PUT SPRITE 3,(X3,Y3),11,3
680 Y3=Y3+D3
690 IF Y3<1 THEN D3=+1
700 IF Y3>174 THEN D3=-1
710 NEXT

```

行番号 110~260

4つのスプライト・パターンを定義

行番号 510~530

4つのスプライトの始動位置と移動方向を設定

行番号 540~710

4つのスプライトを移動させる

①スプライト：「動画」と訳すこともある。MSXで使われている画面処理用ICの特徴的な機能で、1コマの絵のパターンをスプライトとして定義すれば、たやすく画面内を移動させられる。

②MSX-BASIC：MSXに内蔵されているBASIC。アメリカのマイクロソフト社製。

③行番号：BASICの1行1行の最初についている番号。BASICはこの番号を目安に実行・制御・編集の各作業をする。

④メモリ：コンピュータがプログラムやデータを格納しておく記憶装置。メモリの大きさによって記憶できる量も変化する。2章1.4(14ページ)参照。

⑤USR関数：BASICの命令の1つ。BASICからマシン語のプロ

らも BASICのかたちをしていますが、リスト 1-2 のほうは行番号⁽³⁾ 100 ~ 160 でメモリにマシン語プログラムを記憶させ、行番号 200 の **USR**関数⁽⁵⁾で実行しています。行番号 1000 ~ 1420 の DATA 文が実はマシン語のプログラムなのです。

実際に MSX でリスト 1-1 のプログラムを入力・実行してみてください。4 つのスプライトが上下左右に動くのが見られるはずです。次にリスト 1-2 のプログラムを入力してみましょう。ただし、入力し終わって **RUN** する前に入力したプログラムを念入りにチェックしてく

ださい。暴走するおそれがあります。⁽⁶⁾

入力ミスがないことを確認したら実行してみましょう。正しく入力させていけば、4 つのスプライトがものすごい速さで上下左右に動くはずで

す。なぜ BASIC とマシン語ではこんなに速さが違うのでしょうか。少し詳しく考えてみましょう。

ごぞんじの通り、コンピュータは命令されて初めて動作する機械です。命令するのは人間ですし、人間の命令がなければコンピュータはいつまでも黙ったままで、何もしません。人間は、コンピュータにさせたい仕事をコンピュータが

リスト 1-2 リスト 1-1 と同じ動作をするマシン語プログラム

```
100 CLEAR 100,&HD300
110 DEF USR=&HD300
120 I=&HD300
140 READ A$:IF A$="*" THEN 200
150 POKE I,VAL("&H"+A$):I=I+1
160 GOTO 140
200 A=USR(0)
210 END
1000 DATA 3A,AF,FC,32,3F,D4,21,E0
1010 DATA F3,7E,E6,FC,F6,02,77,CD
1020 DATA 72,00,AF,21,BF,D3,CD,B3
1030 DATA D3,3E,01,21,DF,D3,CD,B3
1040 DATA D3,3E,02,21,FF,D3,CD,B3
1050 DATA D3,3E,03,21,1F,D4,CD,B3
1060 DATA D3,21,10,27,E5,AF,21,44
1070 DATA D4,CD,AA,D3,06,1C,0E,D5
1080 DATA 21,45,D4,11,40,D4,CD,9A
1090 DATA D3,3E,01,21,48,D4,CD,AA
1100 DATA D3,06,1C,0E,D5,21,49,D4
1110 DATA 11,41,D4,CD,9A,D3,3E,02
1120 DATA 21,4C,D4,CD,AA,D3,06,01
1130 DATA 0E,AF,21,4C,D4,11,42,D4
1140 DATA CD,9A,D3,3E,03,21,50,D4
1150 DATA CD,AA,D3,06,01,0E,AF,21
1160 DATA 50,D4,11,43,D4,CD,9A,D3
1170 DATA CD,B7,00,E1,38,05,2B,7C
1180 DATA B5,20,A1,3A,3F,D4,CD,5F
1190 DATA 00,C9,1A,86,77,B8,30,04
1200 DATA 3E,01,18,04,B9,D8,3E,FF
```

```
1210 DATA 12,C9,E5,CD,87,00,01,04
1220 DATA 00,18,07,E5,CD,84,00,01
1230 DATA 20,00,EB,E1,C3,5C,00,01
1240 DATA 03,07,0F,1F,3E,7C,F8,F8
1250 DATA 7C,3E,1F,0F,07,03,01,80
1260 DATA C0,E0,F0,F8,7C,3E,1F,1F
1270 DATA 3E,7C,F8,F0,E0,C0,80,01
1280 DATA 02,04,08,18,3C,7E,FF,FF
1290 DATA 7E,3C,18,08,04,02,01,80
1300 DATA 40,20,10,18,3C,7E,FF,FF
1310 DATA 7E,3C,18,10,20,40,80,01
1320 DATA 02,04,08,11,23,47,8F,8F
1330 DATA 47,23,11,08,04,02,01,80
1340 DATA 40,20,10,88,C4,E2,F1,F1
1350 DATA E2,C4,88,10,20,40,80,01
1360 DATA 03,07,0F,1F,27,43,81,81
1370 DATA 43,27,1F,0F,07,03,01,80
1380 DATA C0,E0,F0,F8,E4,C2,81,81
1390 DATA C2,E4,F8,F0,E0,C0,80,00
1400 DATA 01,FF,01,FF,2E,32,00,01
1410 DATA 82,C8,04,0F,0D,4E,08,07
1420 DATA 9B,A2,0C,0B,*
```

行番号 100 ~ 210

マシン語プログラムをメモリに格納し **USR** 関数で実行

行番号 1000 ~ 1420

マシン語プログラムを DATA 文にしたもの

グラムを直接実行させることができる。

①なぜかという、マシン語は 1 つの入力ミスがあっても正しく動作してくれないからである。CPU はまちがって入力したマシン語命令をまちがいとも知らず正確に実行するため、思いがけない動作、たとえばキー入力ができなくなったり画面におかしな文字が表われたりし、場合によってはまったく

入力したプログラムが壊れたりする。こういう状態を「暴走した」という。暴走してしまったら、電源を入れなおす（当然 RAM 上のプログラムは失われる）しかない。

理解できるような命令の並びに替えて、実行させなければなりません。

そもそもコンピュータが理解できる命令はマシン語だけで、マシン語以外のコンピュータ言語、たとえばBASICなどは、そのままではコンピュータにも何の指令だかわからないのです。そこで、「BASIC」などで書かれた命令をコンピュータにもわかるように「マシン語」に翻訳しなければなりません。

この翻訳プログラムは、翻訳のしかたによって大きく2つに分けられ、それぞれ**インタプリタ**と**コンパイラ**と呼ばれています。「インタプリタ」は、命令を1つ1つマシン語に翻訳しながら実行してゆく方式、「コンパイラ」はまず全体を翻訳し、できあがったマシン語プログラムを一気に実行する方式、と覚えておいてください。

BASICのように「翻訳」が必要な言語は、⁽⁷⁾ひとつの命令ごとに多くのマシン語命令に置き換えなければならないため、どうしても直接マシン語命令で書くより命令数が多くなり、当然スピードも落ちます。しかも方式が「インタプリタ」なら、命令の1つ1つで辞書を引くようにして翻訳・実行を繰り返すので、さらに遅くなります。⁽⁸⁾

MSXは、「MSX-BASIC」の文法にそって書かれたBASICプログラムをインタプリタ方式で翻訳して実行するような「MS

X-BASICインタプリタ」を装備しているため、マシン語しか理解できないコンピュータにもBASICプログラムを実行させることができるというわけです。

たとえば、

X0=X0+1

という1つのBASIC命令を実行するときBASICインタプリタは数10、数100行のマシン語命令に置きかえられるので、数m秒(1000分の数秒)の実行時間がかかります。しかし同じものをマシン語命令でつくればわずか3命令程度ですから、⁽⁹⁾10μ秒(100万分の10秒)前後で実行できます。

これまでにマシン語プログラムについてわかった長所・短所をまとめてみましょう。

●長所 ①BASIC でつくったプログラムよりマシン語の方が数10倍実行速度が速い。

●短所 ①BASIC プログラムはミスがあっても暴走することはないが、⁽¹⁰⁾マシン語プログラムにミスがあると暴走することがある。

そのほか、一般にマシン語についていわれていることとして、

●長所 ②メモリの使用量がBASICに比べてかなり小さくなる。⁽¹¹⁾

●短所 ②マシン語命令で使える数値は整数のみ。

③マシン語そのものがBASICなど(高

⑦このような言語のことを「高級言語」という。それに対するマシン語は低級言語である。この差は言語自体の機能差によるものでなく、単に人間にとって理解しやすい言語であるかどうかによる。

⑧いくら「遅くなる」とはいえ、よほど複雑な作業でない限り、人間がするより速い。

⑨ただマシン語命令の中では小数を直接扱うことができないので、必要であればその機能を別のプログラムで用意しなければならない。そのプログラムはやはり数10、数100行にわた

ってしまう。マシン語でプログラムをつくる時は使う数を整数型にすることで高速度の処理ができるようになる。詳細は2章2(18ページ)参照。

⑩もちろん、BASICのインタプリタはマシン語で書かれているので、インタプリタ自体に異常があれば暴走する可能性はある。BASICインタプリタがしっかりしている限りBASICプログラムが暴走することはない。

⑪BASICインタプリタ自体がかなり大規模なプログラムにならざるを得ないこともある。

級言語)に比べて難解。

④マシン語のプログラムをつくるには経験によるノウハウ⁽¹²⁾が必要なため、初心者にはとっつきにくい。

⑤MSX (に限らず現在の安価なパソコン)はマシン語用にはつくられていないため、モニタ⁽¹³⁾を装備していなかった

り、あっても機能が貧弱。

以上、短所と長所を簡単に述べました。これを見る限り短所の方が多いようですが、プログラムの高速化・省メモリ化⁽¹⁴⁾をしようとすればマシン語によるプログラムに勝るものはありません。



1 マシン語のための予備知識

マシン語の表わしかた

コンピュータにはマシン語しかわからないということをお話ししました。BASICで書かれたプログラムは翻訳してやらなければならないということも理解していただけたと思います。

問題は、インタープリタは遅い、という事実です。そこで、スピードを求められるプログラムにはマシン語命令を使わなければならないのです。

ところがこんどは、「コンピュータにとってわかりやすい」ということが「人間にとって難解」である、という問題にぶつかります。コンピュータのマシン語命令は、1つ1つが1つずつの数値に割り当てられていて、CPU (中央処理装置)⁽¹⁵⁾の種類によってどの数値がどの命令を表わすのかが決まっています。コンピュータは、数値を内部で電氣的な「ON / OFF」に

しているため、かえって数値だけで何もかも表現した方が理解しやすいのですが、人間の思考はそのようにできていません。人間にも通じるように、「ON / OFF」信号を「1 / 0」の数に置きかえた2進数、それを4桁ずつまとめた16進数⁽¹⁶⁾などいろんな表現法が考えられたものの、結局すべて数字であるため、ひと目で命令内容を把握することができません。

たとえばMSXのCPU (Z80A)⁽¹⁷⁾では、16進数で「80」はレジスタA⁽¹⁸⁾にレジスタBを加算する命令、同じく「81」はレジスタAにレジスタCを加算する命令、というぐあいに696種類の命令に数値が割り当てられています。でも、「80」と「81」との間にこうした意味の差に見合った違いがあるでしょうか。696種類もの命令をこうした対応のしかたですべて覚えようと

⑫過去に蓄積された実地的な技術。いわゆるコツのようなもの。

⑬直接使用者がマシン語を操作するためのプログラム。メモリの内容を表示・変更したり、セーブ・ロードしたりできる。

⑭マシン語のプログラムは1つ1つの命令が短いうえ、メモリ上にすきまなく詰めることができる。

⑮CPU: コンピュータの中心となって、演算、制御などをする。メモリや周辺機器もCPUの管理下におかれる。2章1. 3 (13ページ) 参照。

⑯われわれが普通に使う10進数以外の数表現については、2章2. 1 (18ページ) 参照。

⑰Z80A: アメリカのザイログ社が開発したCPUのZ80の高速版。全CPUのうちで世界一よく使われ、源流の8080を含めて一般に“80系”と呼ばれている。

⑱レジスタ: CPUの中にある、一時記憶の箱のようなもの。Z80Aには大小あわせて22本のレジスタがある。詳しくは3章1. 1 (35ページ) 参照。

するのは、どだい無理な話です。

そこで、マシン語を人間にもわかりやすいような意味のある「記号」に置きかえることが考えられました。この記号のことをニモニックといいます。さきほどの例に挙げた「レジスタAの値にレジスタBの値を加算して結果をレジスタAの新しい値とする」という（Z80Aの）命令「80」は、ニモニックでは、

ADD A, B

となります。“ADD”とは加算のことで、“A、B”はレジスタAの値にレジスタBの値を加算するということです。ただ「80」と書くよりも、この方がずっとわかりやすいと思いませんか。一般に、マシン語でプログラムをつくるときはこのニモニックを使います。

ニモニックを使ってプログラムを書き、それをマシン語の数値（主に16進数）に置きかえることをアセンブルといいます。アセンブルを手作業でするのがハンド・アセンブルです。実際に経験してみればわかりますが、ハンド・アセ

リスト 1-3

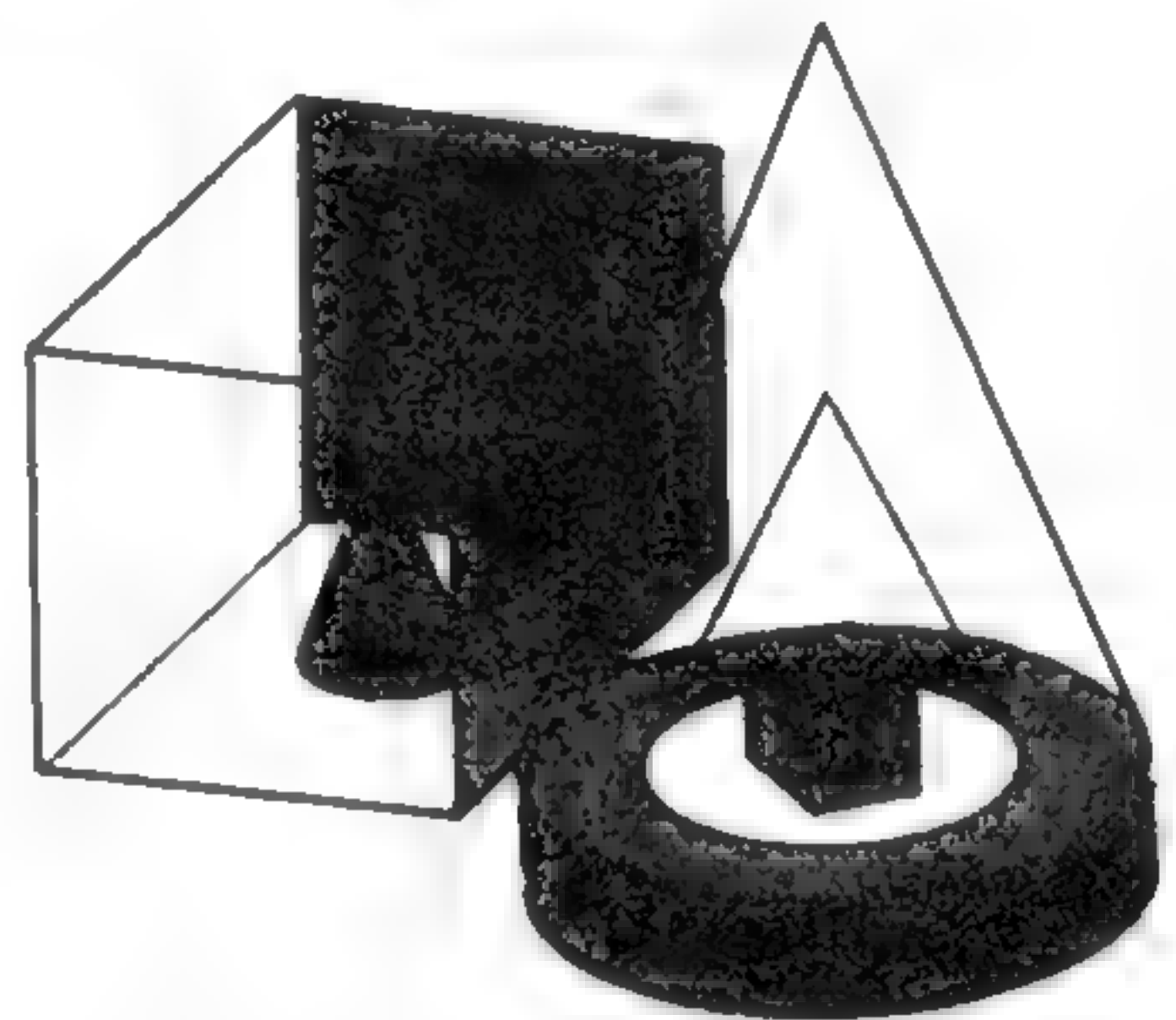
1000:		:
1010:		; Sample Program No. 1
1020:		; (Machine Language)
1030:		:
1040:	D300	ORG 0D300H
1050:		:
1060:		; ** MSX ROMOS ENTRY **
1070:		:
1080:	005C =	LDIRVM EQU 005CH
1090:	005F =	CHGMOD EQU 005FH
1100:	0072 =	INIGRP EQU 0072H
1110:	0084 =	CALPAT EQU 0084H
1120:	0087 =	CALATR EQU 0087H
1130:	00B7 =	BREAKX EQU 00B7H
1140:		:
1150:		; ** BASIC WORK ADDRESS **
1160:		:
1170:	F3E0 =	RG1SAV EQU 0F3E0H
1180:	FCAF =	SCRMOD EQU 0FCAFH
1190:		:
1200:		; ** MAIN **
1210:		:
1220:	D300	START:
1230:	D300 3AAFFC	LD A, (SCRMOD)
1240:	D303 323FD4	LD (SCMD), A
1250:		
1260:	D306 21E0F3	LD HL, RG1SAV
1270:	D309 7E	LD A, (HL)
1280:	D30A E6FC	AND 0FCH
1290:	D30C F602	OR 2
1300:	D30E 77	LD (HL), A
1310:	D30F CD7200	CALL INIGRP
1320:		:
1330:	D312 AF	XOR A
1340:	D313 21BFD3	LD HL, SPRPT0
1350:	D316 CDB3D3	CALL SPRSET

ンブルは実に非能率的でまちがいやすく、慎重になればなるほど煩雑で面倒な仕事です。せっかくコンピュータに作業をさせて楽しようとしているのに、その前に人間がこんなことで苦勞しなければならないのはなぜだろう、とつい考えてしまうほどです。そこでそのアセンブル作業をもコンピュータにさせてしまおうというプログラムが考えられました。そのプログラムがアセンブラです。

リスト1-3はリスト1-2のマシン語プロ

グラム部分をニモニクで記述し、アセンブラによってアセンブルしたものです。ニモニクとマシン語については第3章で、アセンブラについては第4章で（MSX用アセンブラのリストも添えて）それぞれ詳しく説明します。

1360:	D319 3E01	LD A,1
1370:	D31B 21DFD3	LD HL,SPRPT1
1380:	D31E CDB3D3	CALL SPRSET
1390:	D321 3E02	LD A,2
1400:	D323 21FFD3	LD HL,SPRPT2
1410:	D326 CDB3D3	CALL SPRSET
1420:	D329 3E03	LD A,3
1430:	D32B 211FD4	LD HL,SPRPT3
1440:	D32E CDB3D3	CALL SPRSET
1450:		;
1460:	D331 211027	LD HL,10000
1470:	D334	LOOP:
1480:	D334 E5	PUSH HL
1490:		
1500:	D335 AF	XOR A
1510:	D336 2144D4	LD HL,SPTAT0
1520:	D339 CDAAD3	CALL SPRMOV
1530:	D33C 061C	LD B,28
1540:	D33E 0ED5	LD C,212+1
1550:	D340 2145D4	LD HL,X0
1560:	D343 1140D4	LD DE,D0
1570:	D346 CD9AD3	CALL POSUPD
1580:	D349 3E01	LD A,1
1590:	D34B 2148D4	LD HL,SPTAT1
1600:	D34E CDAAD3	CALL SPRMOV
1610:	D351 061C	LD B,28
1620:	D353 0ED5	LD C,212+1
1630:	D355 2149D4	LD HL,X1
1640:	D358 1141D4	LD DE,D1
1650:	D35B CD9AD3	CALL POSUPD
1660:	D35E 3E02	LD A,2
1670:	D360 214CD4	LD HL,SPTAT2
1680:	D363 CDAAD3	CALL SPRMOV
1690:	D366 0601	LD B,1
1700:	D368 0EAF	LD C,174+1
1710:	D36A 214CD4	LD HL,Y2
1720:	D36D 1142D4	LD DE,D2




```

1730: D370 CD9AD3 CALL POSUPD
1740: D373 3E03 LD A,3
1750: D375 2150D4 LD HL,SPTAT3
1760: D378 CDAAD3 CALL SPRMOV
1770: D37B 0601 LD B,1
1780: D37D 0EAF LD C,174+1
1790: D37F 2150D4 LD HL,Y3
1800: D382 1143D4 LD DE,D3
1810: D385 CD9AD3 CALL POSUPD
1820:
1830: D388 CDB700 CALL BREAKX
1840: D38B E1 POP HL
1850: D38C 3805 JR C,BRKEND
1860: D38E 2B DEC HL
1870: D38F 7C LD A,H
1880: D390 B5 OR L
1890: D391 20A1 JR NZ,LOOP
1900: ;
1910: D393 BRKEND:
1920: D393 3A3FD4 LD A,(SCMD)
1930: D396 CD5F00 CALL CHGMOD
1940: D399 C9 RET
1950: ;
1960: ; ** SPRITE POSITION UPDATE **
1970: ;
1980: D39A POSUPD:
1990: D39A 1A LD A,(DE)
2000: D39B 86 ADD A,(HL)
2010: D39C 77 LD (HL),A
2020: D39D B8 CP B
2030: D39E 3004 JR NC,POSUP1
2040: D3A0 3E01 LD A,1
2050: D3A2 1804 JR POSUP2
2060: D3A4 POSUP1:
2070: D3A4 B9 CP C
2080: D3A5 D8 RET C
2090: D3A6 3EFF LD A,-1
2100: D3A8 POSUP2:
2110: D3A8 12 LD (DE),A
2120: D3A9 C9 RET
2130: ;
2140: ; ** SPRITE MOVE **
2150: ;
2160: D3AA SPRMOV:
2170: D3AA E5 PUSH HL
2180: D3AB CD8700 CALL CALATR
2190: D3AE 010400 LD BC,4
2200: D3B1 1807 JR VRMOV
2210: ;
2220: ; ** SPRITE PATTERN SET **
2230: ;
2240: D3B3 SPRSET:

```



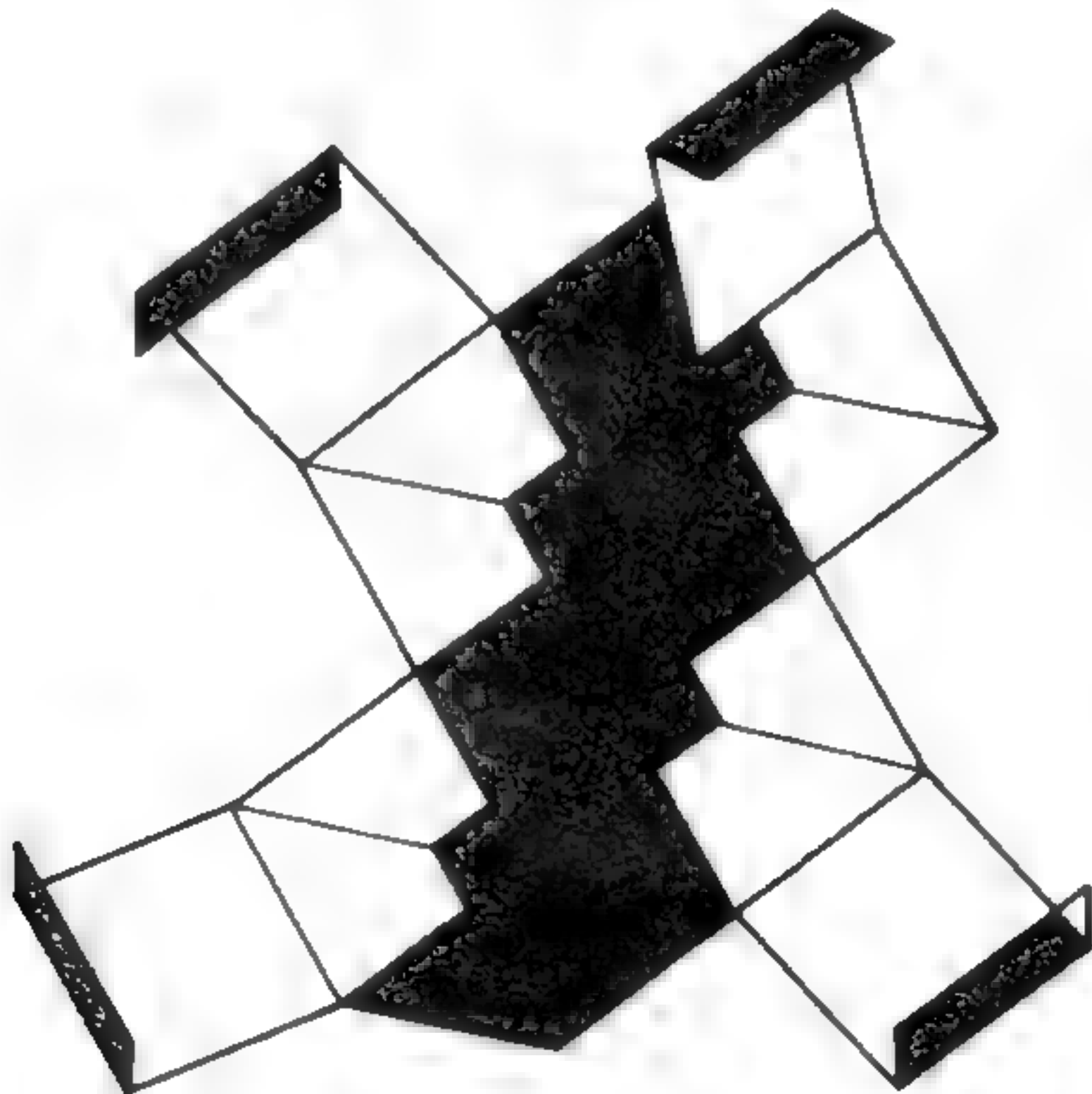
```

2250: D3B3 E5          PUSH HL
2260: D3B4 CD8400       CALL CALPAT
2270: D3B7 012000       LD BC,32
2280: D3BA              VRMOV:
2290: D3BA EB          EX DE,HL
2300: D3BB E1          POP HL
2310: D3BC C35C00       JF LDIRVM
2320:                  ;
2330:                  ; ** SPRITE PATTERN **
2340:                  ;
2350: D3BF              SPRPT0:
2360: D3BF 0103070F     DEFB 001H,003H,007H,00FH
2370: D3C3 1F3E7CF8     DEFB 01FH,03EH,07CH,0F8H
2380: D3C7 F87C3E1F     DEFB 0F8H,07CH,03EH,01FH
2390: D3CB 0F070301     DEFB 00FH,007H,003H,001H
2400: D3CF 80C0E0F0     DEFB 080H,0C0H,0E0H,0F0H
2410: D3D3 F87C3E1F     DEFB 0F8H,07CH,03EH,01FH
2420: D3D7 1F3E7CF8     DEFB 01FH,03EH,07CH,0F8H
2430: D3DB F0E0C080     DEFB 0F0H,0E0H,0C0H,080H
2440: D3DF              SPRPT1:
2450: D3DF 01020408     DEFB 001H,002H,004H,008H
2460: D3E3 183C7EFF     DEFB 018H,03CH,07EH,0FFH
2470: D3E7 FF7E3C18     DEFB 0FFH,07EH,03CH,018H
2480: D3EB 08040201     DEFB 008H,004H,002H,001H
2490: D3EF 80402010     DEFB 080H,040H,020H,010H
2500: D3F3 183C7EFF     DEFB 018H,03CH,07EH,0FFH
2510: D3F7 FF7E3C18     DEFB 0FFH,07EH,03CH,018H
2520: D3FB 10204080     DEFB 010H,020H,040H,080H
2530: D3FF              SPRPT2:
2540: D3FF 01020408     DEFB 001H,002H,004H,008H
2550: D403 1123478F     DEFB 011H,023H,047H,08FH
2560: D407 8F472311     DEFB 08FH,047H,023H,011H
2570: D40B 08040201     DEFB 008H,004H,002H,001H
2580: D40F 80402010     DEFB 080H,040H,020H,010H
2590: D413 88C4E2F1     DEFB 088H,0C4H,0E2H,0F1H
2600: D417 F1E2C488     DEFB 0F1H,0E2H,0C4H,088H
2610: D41B 10204080     DEFB 010H,020H,040H,080H
2620: D41F              SPRPT3:
2630: D41F 0103070F     DEFB 001H,003H,007H,00FH
2640: D423 1F274381     DEFB 01FH,027H,043H,081H
2650: D427 8143271F     DEFB 081H,043H,027H,01FH
2660: D42B 0F070301     DEFB 00FH,007H,003H,001H
2670: D42F 80C0E0F0     DEFB 080H,0C0H,0E0H,0F0H
2680: D433 F8E4C281     DEFB 0F8H,0E4H,0C2H,081H
2690: D437 81C2E4F8     DEFB 081H,0C2H,0E4H,0F8H
2700: D43B F0E0C080     DEFB 0F0H,0E0H,0C0H,080H
2710:                  ;
2720:                  ; ** WORK AREA **
2730:                  ;
2740: D43F              SCMD: DEFS 1
2750:
2760: D440 01          D0: DEFB +1

```


2770:	D441 FF	D1: DEFB -1
2780:	D442 01	D2: DEFB +1
2790:	D443 FF	D3: DEFB -1
2800:		
2810:	D444	SPTAT0:
2820:	D444 2E	Y0: DEFB 46
2830:	D445 32	X0: DEFB 50
2840:	D446 0001	DEFB 0,1
2850:	D448	SPTAT1:
2860:	D448 82	Y1: DEFB 130
2870:	D449 C8	X1: DEFB 200
2880:	D44A 040F	DEFB 4,15
2890:	D44C	SPTAT2:
2900:	D44C 0D	Y2: DEFB 13
2910:	D44D 4E	X2: DEFB 78
2920:	D44E 0807	DEFB 8,7
2930:	D450	SPTAT3:
2940:	D450 9B	Y3: DEFB 155
2950:	D451 A2	X3: DEFB 162
2960:	D452 0C0B	DEFB 12,11
2970:		;
2980:	D454	END

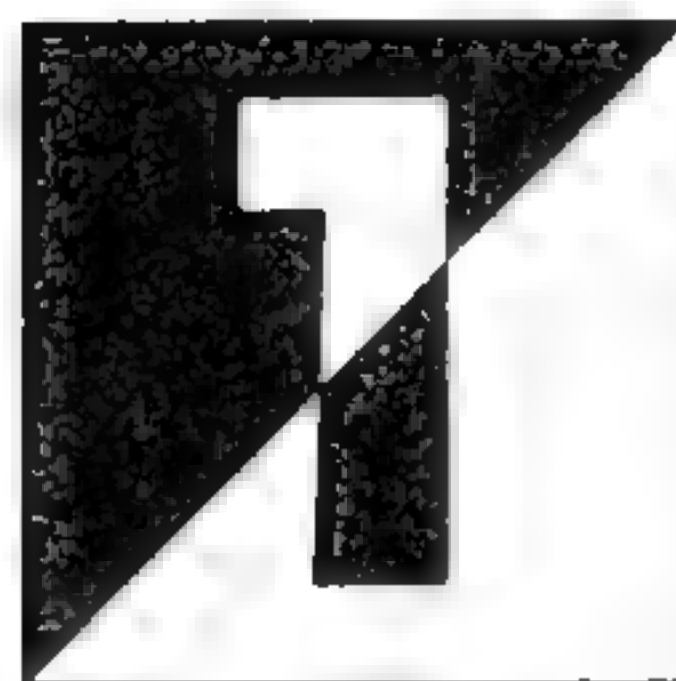
00B7	BREAKX	D393	BRKEND	0087	CALATR	0084	CALPAT
005F	CHGMOD	D440	D0	D441	D1	D442	D2
D443	D3	0072	INIGRP	005C	LDIRVM	D334	LOOP
D3A4	POSUP1	D3A8	POSUP2	D39A	POSUPD	F3E0	RG1SAV
D43F	SCMD	FCAF	SCRMOD	D3AA	SPRMOV	D3BF	SPRPT0
D3DF	SPRPT1	D3FF	SPRPT2	D41F	SPRPT3	D3B3	SPRSET
D444	SPTAT0	D448	SPTAT1	D44C	SPTAT2	D450	SPTAT3
D300	START	D3BA	VRMOV	D445	X0	D449	X1
D44D	X2	D451	X3	D444	Y0	D448	Y1
D44C	Y2	D450	Y3				





2章 基礎知識

1章で説明した通り，マシン語はコンピュータのハードウェアに最も密着したコンピュータ言語です。ですから，コンピュータのハードウェアがどのようなになっているかの知識なしには応用できません。この章では1章で解説した予備知識を元に，ハードウェアやコンピュータ内での数表現などについて，より深く詳細な説明をしてゆきます。



2. 基礎知識

コンピュータの構造

1. ハードウェアとソフトウェア

コンピュータを構成する装置、またはコンピュータ自体をハードウェア (Hardware, 金物という意味) といい、それに対してハードウェアを動かすプログラムのことをソフトウェア (Software, やわものという意味) といいます。

ソフトウェアはハードウェア上でその機能の助けを借りて役を果たすものであるため、ハードウェアの機能上の差はソフトウェアを動作させる環境に大きな影響を及ぼします。たとえば、カセットしかないシステムではディスクにプログラムを記録しておくようなソフトウェアは当然動きません。

しかし、ソフトウェアにもハードウェアに左右されないものがあります。図2-1はいくつかのコンピュータ言語がハードウェアにどの程度頼っ

ているかを示したものです。見てのとおり、高級言語と呼ばれているソフトウェアの中でも、BASIC (マイコン用) はハードウェアに頼っている部分が多く、そのため機種によってはグラフィックや音などの機能拡張部分が異なる場合さえあります。またFORTRANやCOBOLは、ハードウェアに頼っている度合いが少ないためにハードウェアの差異にかかわらず同じように使えます。

それに対してマシン語 (ニモニックなども含む) は、1章2 マシン語の表わしかたの項でも述べたようにCPUごとに命令が異なるので、完全にハードウェア (特にCPU) = マシン語といってもよいのです。

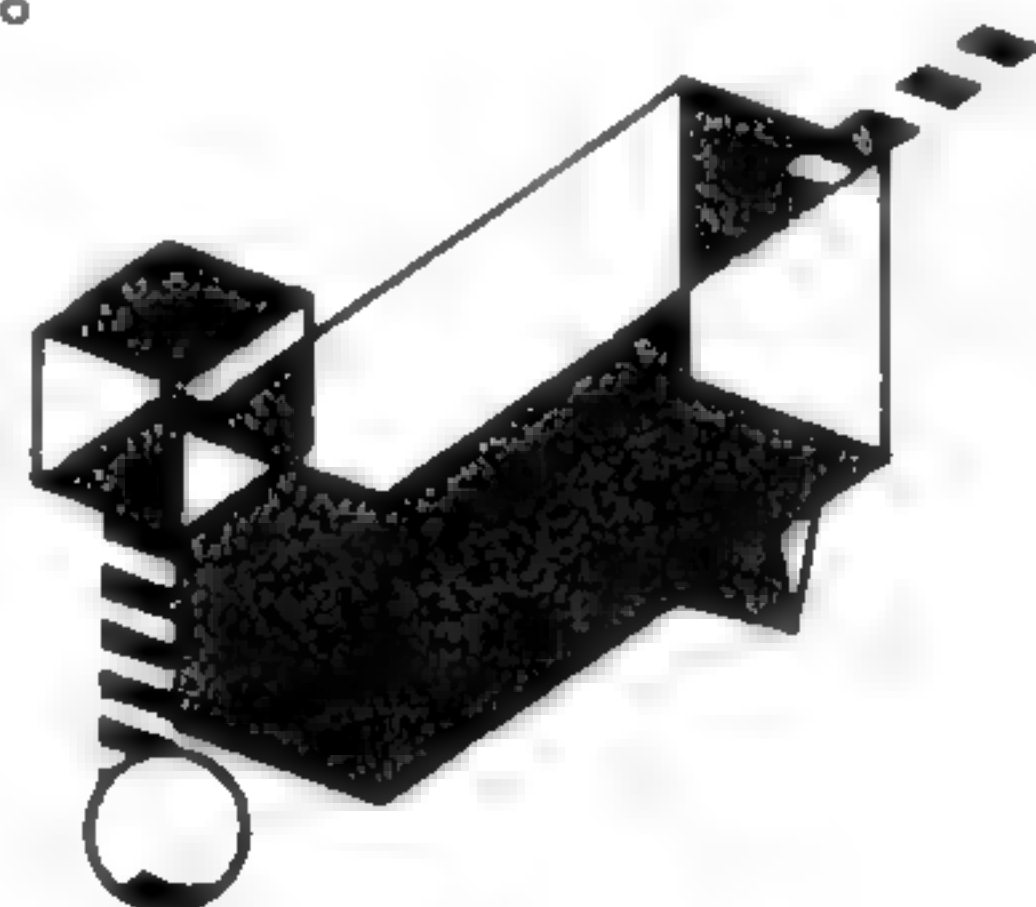
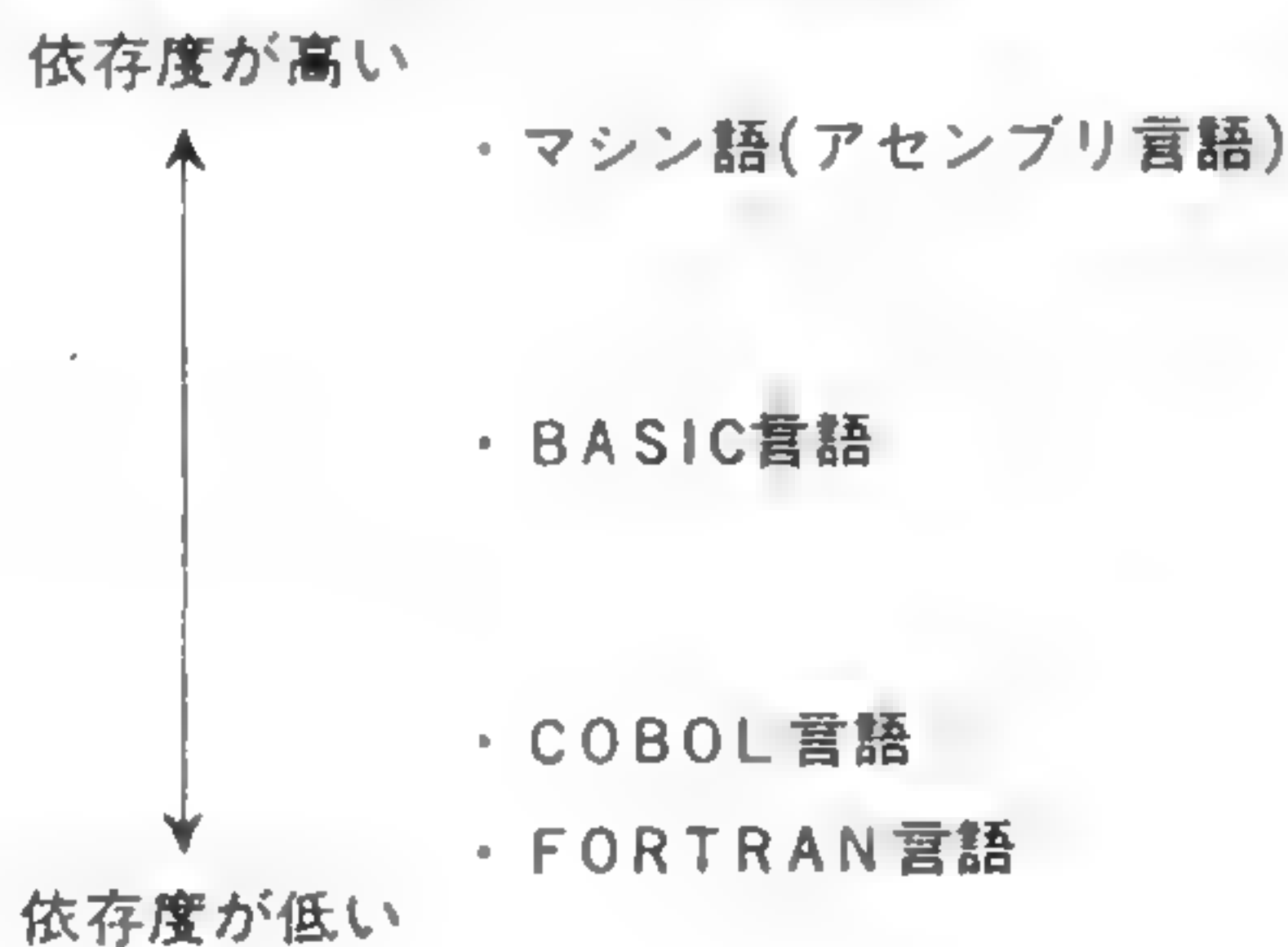


図2-1 言語別のハードウェア依存度



2. コンピュータの構成

コンピュータと呼ばれる機械は、電卓から超大型機まですべて次の5つの機能を備えています。5つの機能とは、入力、出力、記憶、演算、制御です。これらの機能をどのハードウェアが受け

①グラフィック関係のBASIC命令は、いまでは機種による方言があたりまえのようになっている (MSXを除く)。つぎつぎに出た新製品が、前に出た他社製品のグラフィック能力を上回っていることをセールス・ポイントの1つにしていたからである。最も原始的なBASICが備えていなかった命令群 (のちに拡張された機能) は、グラフィック関係に限らず大なり小なりこのような状態である。

②FORTRAN: 米国IBM社が開発した、世界初の高級言語。FO

Rmula TRANslator の略で、数式に近い表現で計算命令を記述できるのが特徴。歴史は古いが、最もよく使われている言語のひとつ。

COBOL: 1960年から、米国政府が中心となって仕様を決めた高級言語。COmmon Business Oriented Language の略で、ビジネス向き言語。

③FORTRAN, COBOLともコンパイラ言語で、コンパイル (高級言語のプログラムをマシン語に一括翻訳すること。詳しく

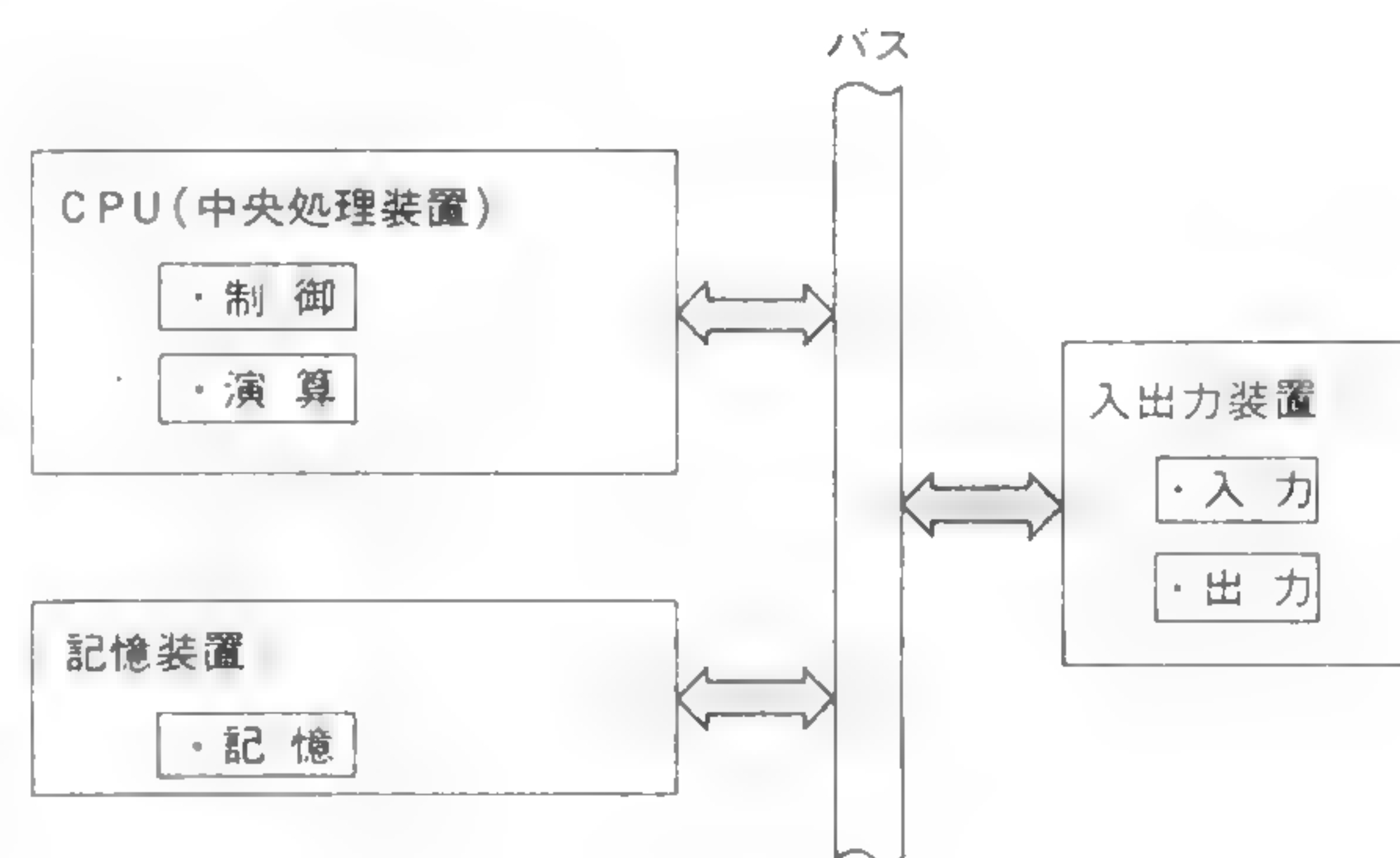
持っているかを図にしたのが図2-2です。ただしこの図はマイコン、パソコン・レベルのコンピュータ²²⁾でのことで、小型計算機以上のマシンでは構成が多少複雑になります。

コンピュータは、制御・演算を受け持つ中央処理装置(CPU)、プログラムやデータを記憶するための記憶装置(メモリ)、外部とのデータのやりとりを受け持つ入出力装置からなります。

これらの装置はバス(Bus)²³⁾と呼ばれる信号線で接続されています。

それでは、これから各装置についてさらに詳しく見ていきましょう。

図2-2 計算機の構成



3. CPU

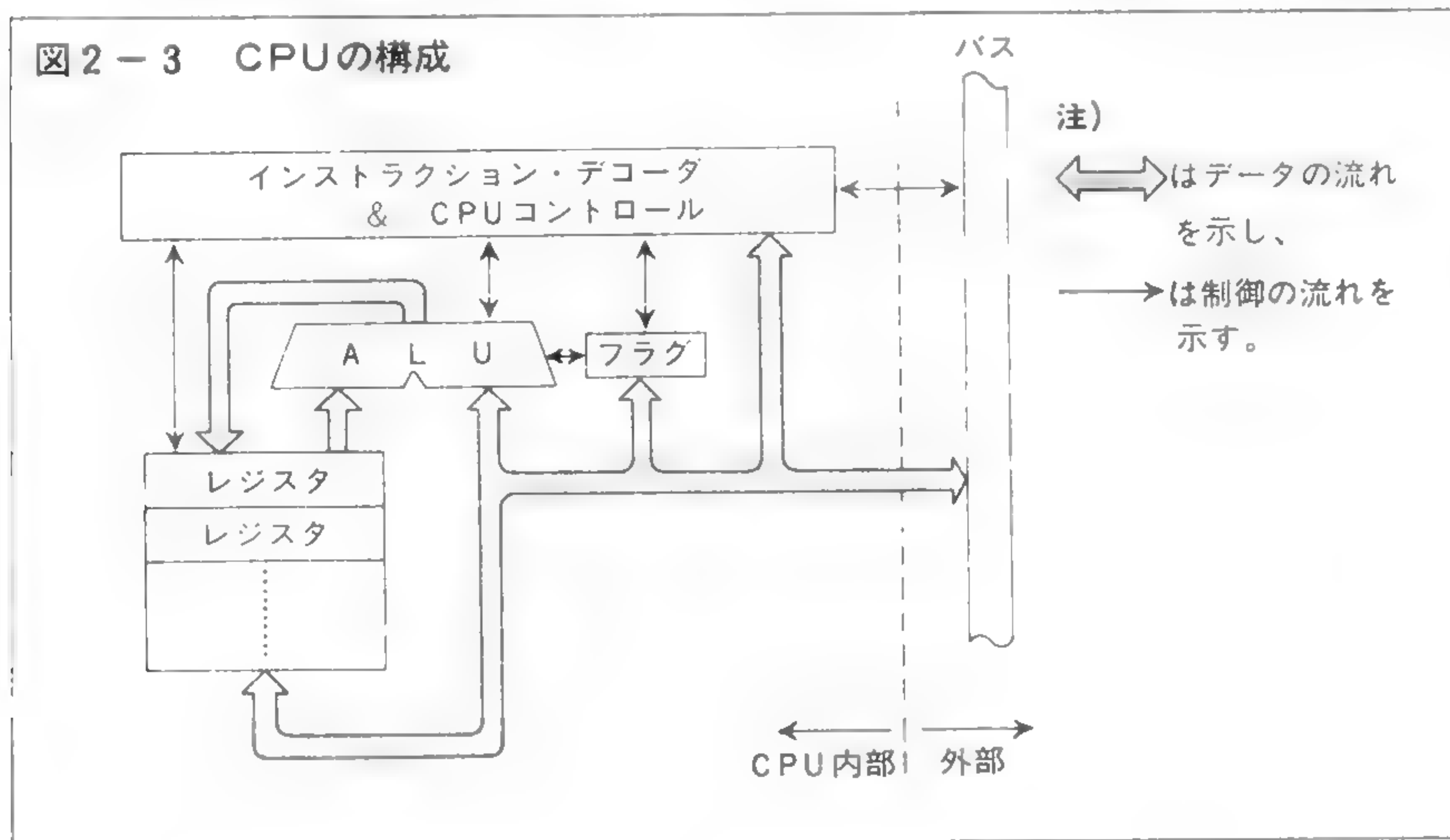
中央処理装置CPU (Central Processing Unit) はコンピュータの中心的な部分で、人間で

いえば頭脳に当たります。CPUの大きな仕事は演算と制御の2つです。図2-3はCPUの構成を示します。

この図でALU (Arithmetic and Logic Unit

= 算術演算装置) は、演算・計算をするところ、レジスタ (Register) はCPU内のデータの一時記憶場所²⁴⁾、インストラクション・デコーダ&CPUコントロール部 (Instruction Decoder & CPU Control) は、CPUに送られた命令を分析してC

図2-3 CPUの構成



注)
 ⇔ はデータの流れを示し、
 → は制御の流れを示す。

は1章1 [2ページ] 参照) してマシン語にするので、コンパイル結果のマシン語(オブジェクト)は他機種との互換性がまったくない。

②小型計算機：マイコンはマイクロ(微細)なコンピュータだから、小型(ミニ)のコンピュータの方が大きい。小型計算機は一般にミニコンと呼ばれる。

③バス：信号の通りみちである信号線を乗合バスにたとえたもの。

④レジスタにも専用の用途を持つものと、何にでも使えるものがある。Aレジスタ、Fレジスタ、SP(スタック・ポインタ)などは用途が限られていて、そのほかのためには使えない。B、C、D、E、H、Lレジスタなどは、計算、アドレスの指定などいろいろな用途に使えるので汎用レジスタなどとも呼ばれる。詳しくは3章1. 1 (35ページ)参照。また、CPUだけが使って、プログラムからは操作できないレジスタ(I、Rなど)もある。

P U 全体の機能をコントロールする部分です。
フラグ(Flag)は、各命令を実行した後の結果がある一定の状態になったかどうかを表わす部分です。²⁴⁾

M S X の Z 8 0 A C P U については、第3章で詳しく説明します。

4. メモリ

記憶装置のメモリ(Memory Unit)は、データやプログラムを記憶するためのハードウェアです。メモリはビットと呼ばれるものの集まりで、ビットはバイト単位に区切られ、その単位でC P U から読み出し(リード)、書き込み(ライト)が行なわれます。

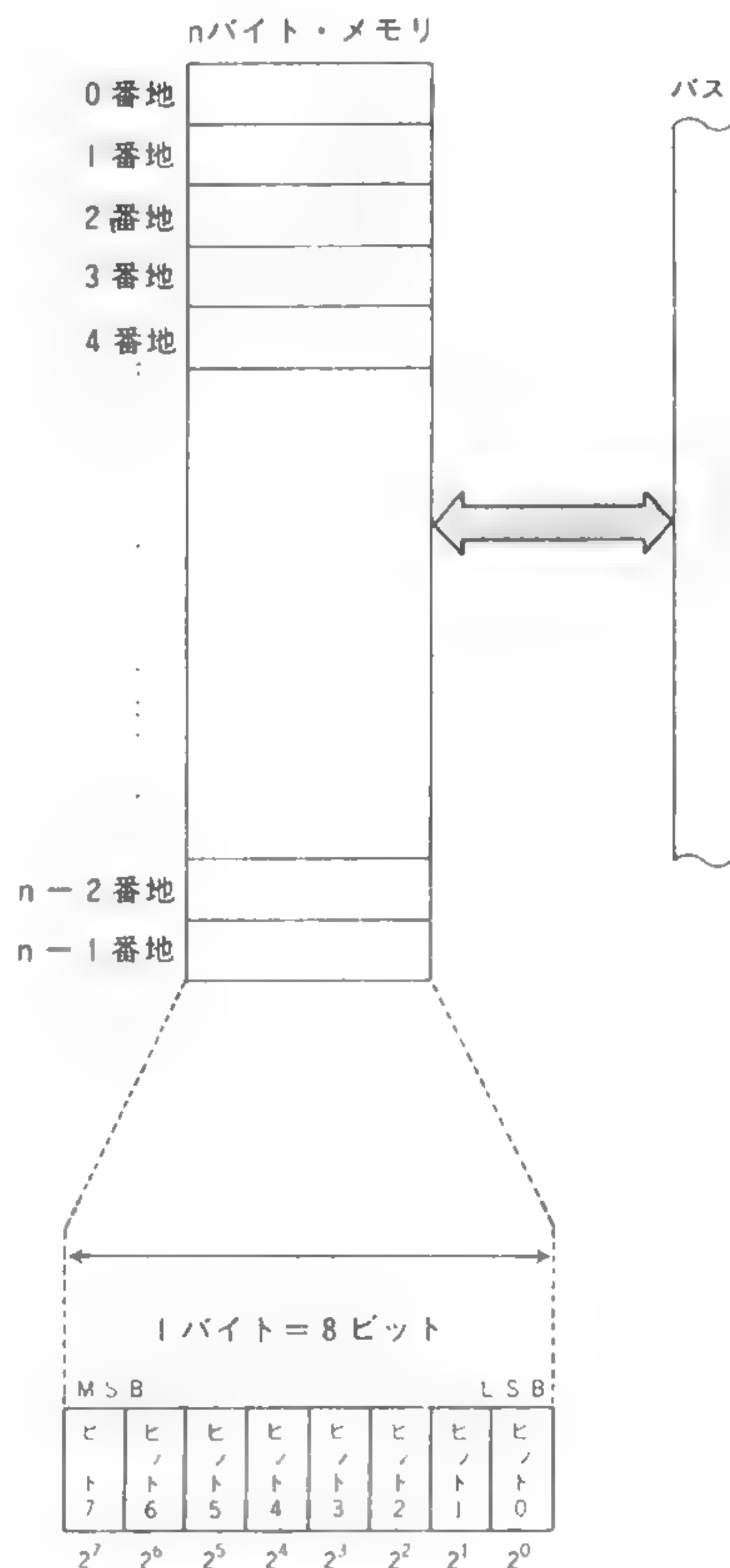
ビット(Bit)とは記憶の最小単位で、1章2マシン語の表わしかたのところで述べたとおり、コンピュータ内部の電氣的な「ON/OFF」を表現しています。したがって1ビットで表わされる値は「0」と「1」の2種類しかありません。ビットを2つ並べたときは0と1の組み合わせが4種類(00, 01, 10, 11)できます。つまり、1ビットで0～1、2ビットで0～3、3ビットで0～7までが表現できるわけです。

バイト(Byte)とは、ビットを8つ並べたもので、表わされる数値は0～255(2^8-1)までです。この「バイト」はメモリの容量を表現する単位にも使われ、メモリの容量を16Kバイトとか32Kバイトとかいいます。²⁵⁾

メモリにはバイト単位に**アドレス**(Address=番地)が割り当てられています(図2-4)。た

例えば16Kバイトのメモリには0番地から16383番地($16 \times 2^{10} - 1$)までのアドレスが割り当てられます。

図2-4 記憶装置の構成



注) ・MSBは「Most significant bit」の略、MSB=ビット7
 ・LSBは「Least significant bit」の略、LSB=ビット0
 ・各ビットは右端よりビット0、ビット1、……ビット7と呼びます。計算機によっては左端よりビット0、ビット1……と呼ぶ場合もあります。

パソコンに使われるメモリICにはRAM(Random Access Memory)とROM(Read Only Memory)があります。²⁷⁾ RAMは、メモリの内容を自由に読み出したり書きかえたりするこ

②4 フラグがまとめてあるレジスタがFレジスタである。3章1.1参照。

②5 一般に「キロ」といえば1000を表わすが、メモリの1キロ(1K)バイトは1K=1024(2^{10})である。Kの上には、M(メ

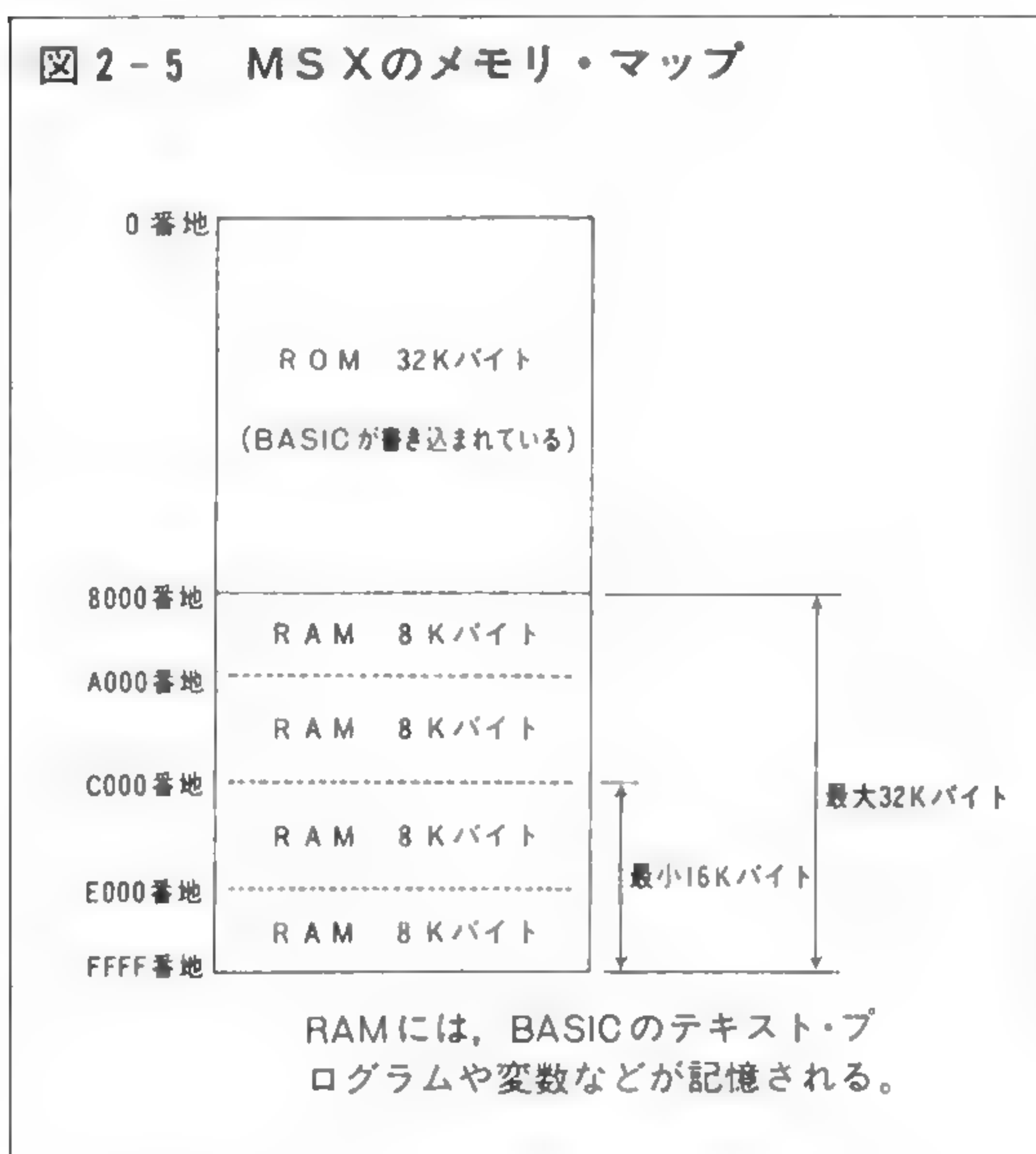
ガ)= 2^{20} 、G(ギガ)= 2^{30} などがある。

②7 ROM、RAMとも、メモリICには1個当たり2K、4K、8K、16K、32Kの容量のものなどが開発されている。

②8 プリンタ：コンピュータの外部出力装置の1つ。コンピュ

とができますが、電源を切ると記憶内容がすべて消えてしまいます。それに対してROMは、名前の通り読み出ししかできないメモリですが、電源を切っても記憶されたデータやプログラムが消えてしまうことはありません。

MSXのZ80A CPUは最大で64Kバイトのメモリを持つことができ、その中でROMとRAMとを混用しています。図2-5のようにメモリの全アドレスがどう使われているかを示すものをメモリ・マップといいます。



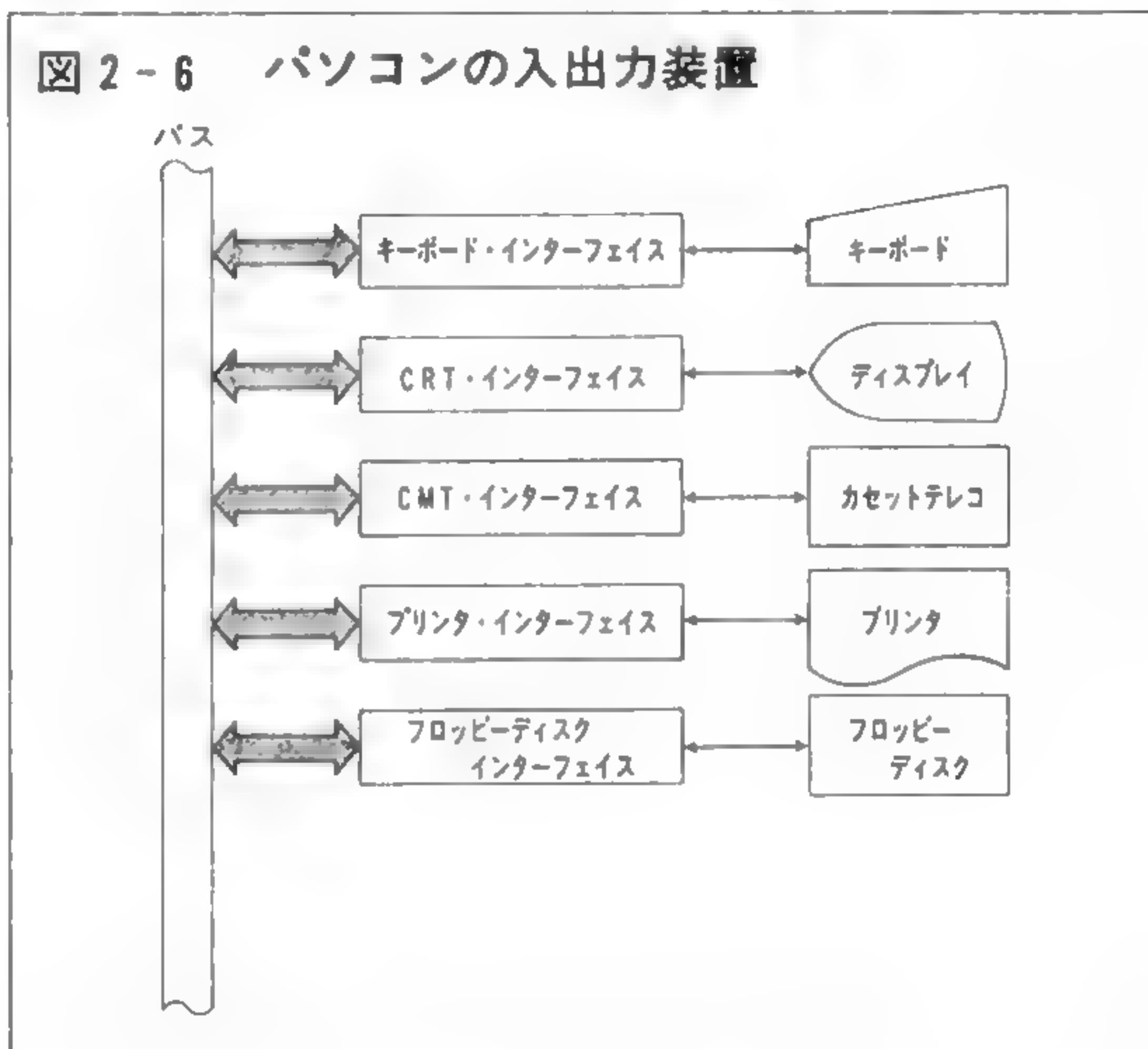
また、メモリにはCPUがリード/ライトしたいアドレスを指定してからリード/ライト完了までの時間であるアクセス・タイム、あるアドレスのリード/ライトを始めてから次のアドレスのリード/ライトを始めるまでに必要な最小時間であるサイクル・タイムなどのタイミングがあります。これらはハードウェアの分野に含まれることですが、コンピュータはこれらの時間が短

ければそれだけ高速に処理ができるということを知っておいてください。

5. 入出力装置

入出力装置 (Input/Output Unit) の構成を図にしたものが図2-6です。

ディスプレイ (画面) やカセット・テレコなど、外部に接続されている装置との情報の受け渡しやコントロールをしているのがI/Oインターフェイス (Input/Output Interface) と呼ばれるものです。I/Oインターフェイスは外部に接続する装置により回路や内部の構成が異なり、たとえば、ディスプレイに使う専用のインターフェイスをカセットに流用するわけにはいきません。あなたがMSXでプリンタを使いたいときにはプリンタ・インターフェイスが必要になるわけです。もちろん、あなたのMSXにプリンタ・インターフェイスが内蔵されていれば、プリンタはケーブルですぐ接続できます。



一タ内の情報を文字や絵にして紙などに印刷することができる。本書に掲載されているリストもプリンタで印字したものである。

●同じMSXでも、発売しているメーカーによってはプリンタ・

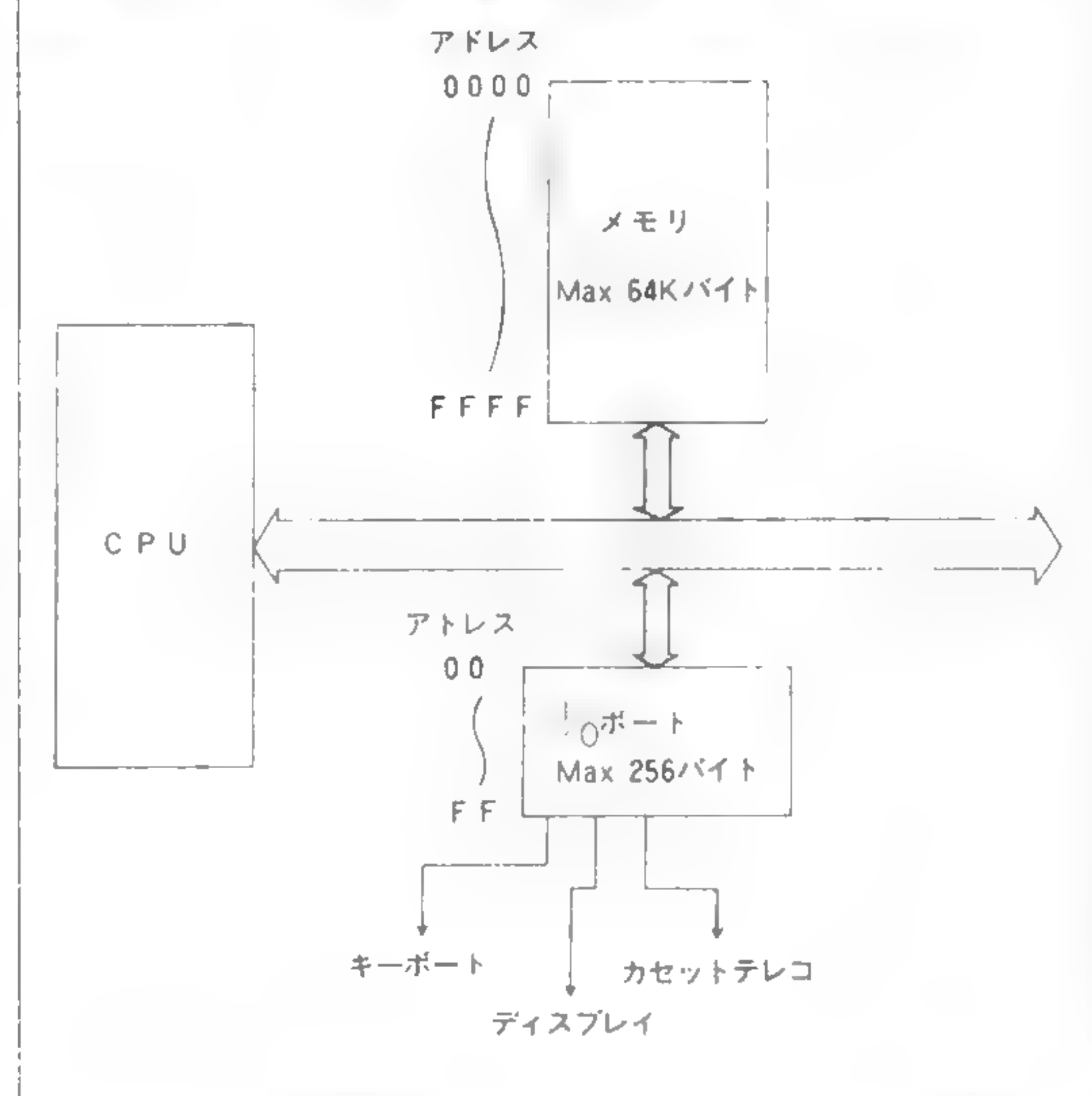
インターフェイスを内蔵しているものと内蔵していないものがある。

Z80A CPUは、メモリとは別に入出力用に使う256バイトの空間を持っています。この空間のことをI/Oポート (I/O Port) または単にポート (Port) と呼びます。ポートとは“港”のことで、外部とデータをやりとりするときの中継点です。ちょうど船で荷物をやりとりするとき、港が陸での最後の中継点となるように、CPUから外部の機器へ送られるデータはこの「ポート」を最後の経由地とします。Z80Aには入出力命令があり、入出力命令を実行するとCPUはポートを中継してデータを入出力します。ポートには1バイト単位に0番地から255番地までのアドレスが割り当てられていて、入出力命令ではこの番地を指定してポートを使います (図2-7)。

各I/Oインターフェイスは最小1ビットから最大数バイトのポートを使いますが、各I/Oイ

ンターフェイスごとに使うアドレスが異なっていないければなりません。たとえば、同じI/Oアドレスを使うインターフェイスは同時に動作させることができません。^{③①}

図2-7 メモリ空間とI/O空間

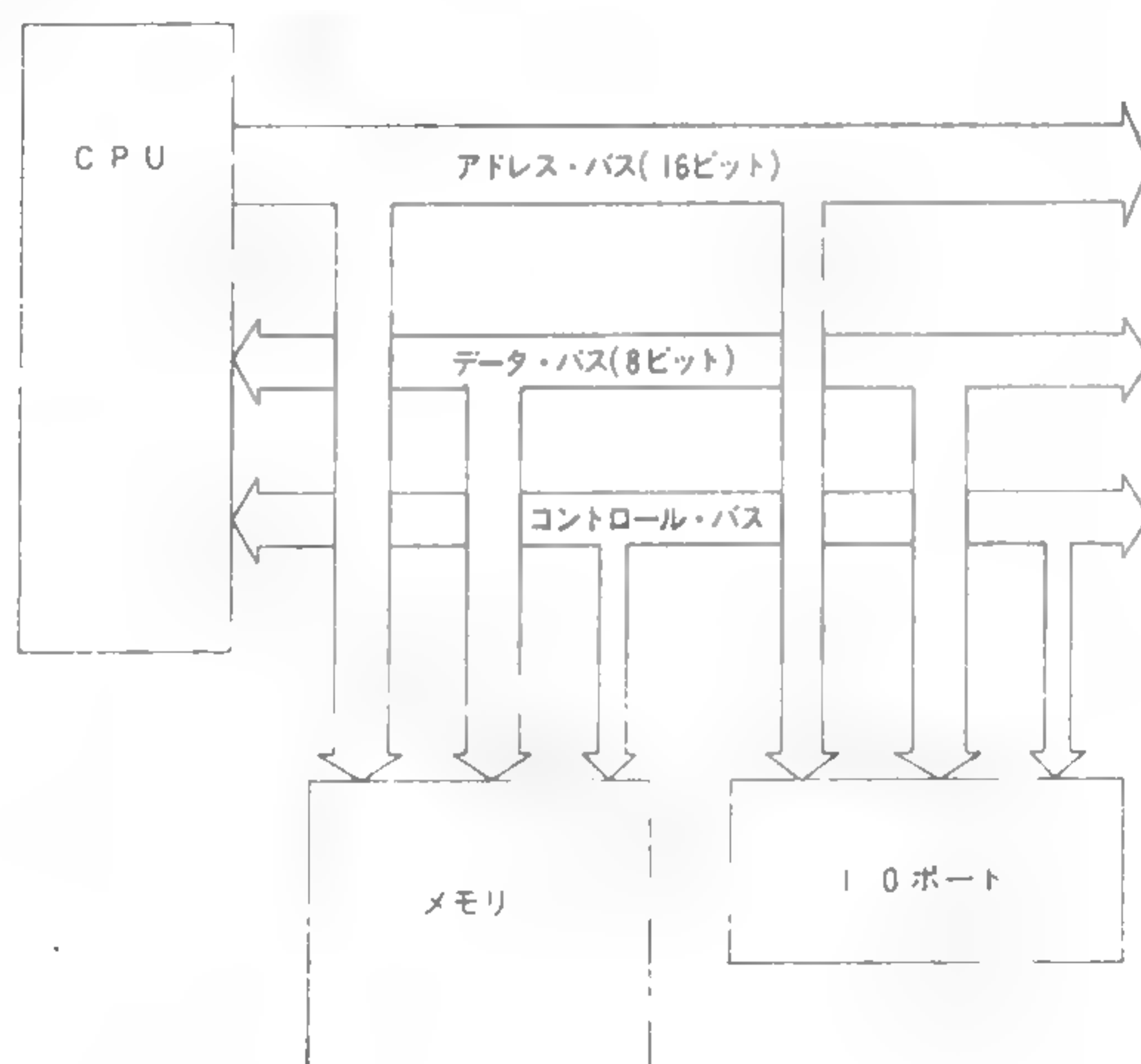


6. バス

バス (Bus) とは、CPU、メモリ、入出力装置などを結び、これらの間でデータのやりとりをするための信号線のことです。

Z80Aには、メモリやI/Oポートのアドレスを送るためのアドレス・バス (Address Bus)、メモリやI/OポートとCPUとの間でデータをやりとりするためのデータ・バス (Data Bus)、メモリや入出力装置、あるいはCPUを制御するた

図2-8 バスの構成



③①自作のインターフェイスなどでは特に注意。もしほかのインターフェイスと同時に動作させると、入出力したデータが変わってしまったり、最悪の場合CPUやI/OインターフェイスのICを破壊してしまったりする。また、自作のインターフェイスは当然ながらMSX BASICのサポートを受けていないので、入出力するプログラムも自作しなければならない。

③①CPUをリセット (初期化) する信号、クロック (CPUの動作の基準となる) 信号、割り込み (1つのプログラムを動作中に強制的にほかのプログラムに制御を移すこと) の要求の信号などがある。

めの情報を送るコントロール・バス (Control Bus) の3種類があります (図2-8)。

①アドレス・バスは、CPUが操作したいメモリやI/Oポートのアドレスを指定するときに使う信号線です。

②データ・バスは、CPUがメモリやI/Oポートにある情報をリード/ライトしたいときにその情報を入出力する信号線です。信号には「CPU→メモリ」「メモリ→CPU」「CPU→I/Oポート」「I/Oポート→CPU」の4種の転送方向があります。

③コントロール・バスには、CPUが出力するデータ・バスの転送方向を指定したり、CPU自体の働きをコントロール⁽³⁾したりする役割があります。

図2-9はCPUがバスをどのように使ってメモリとの間でデータをやりとりしているかを示した図です。これからその動作を説明しますが少し難しいので飛ばしても結構です。

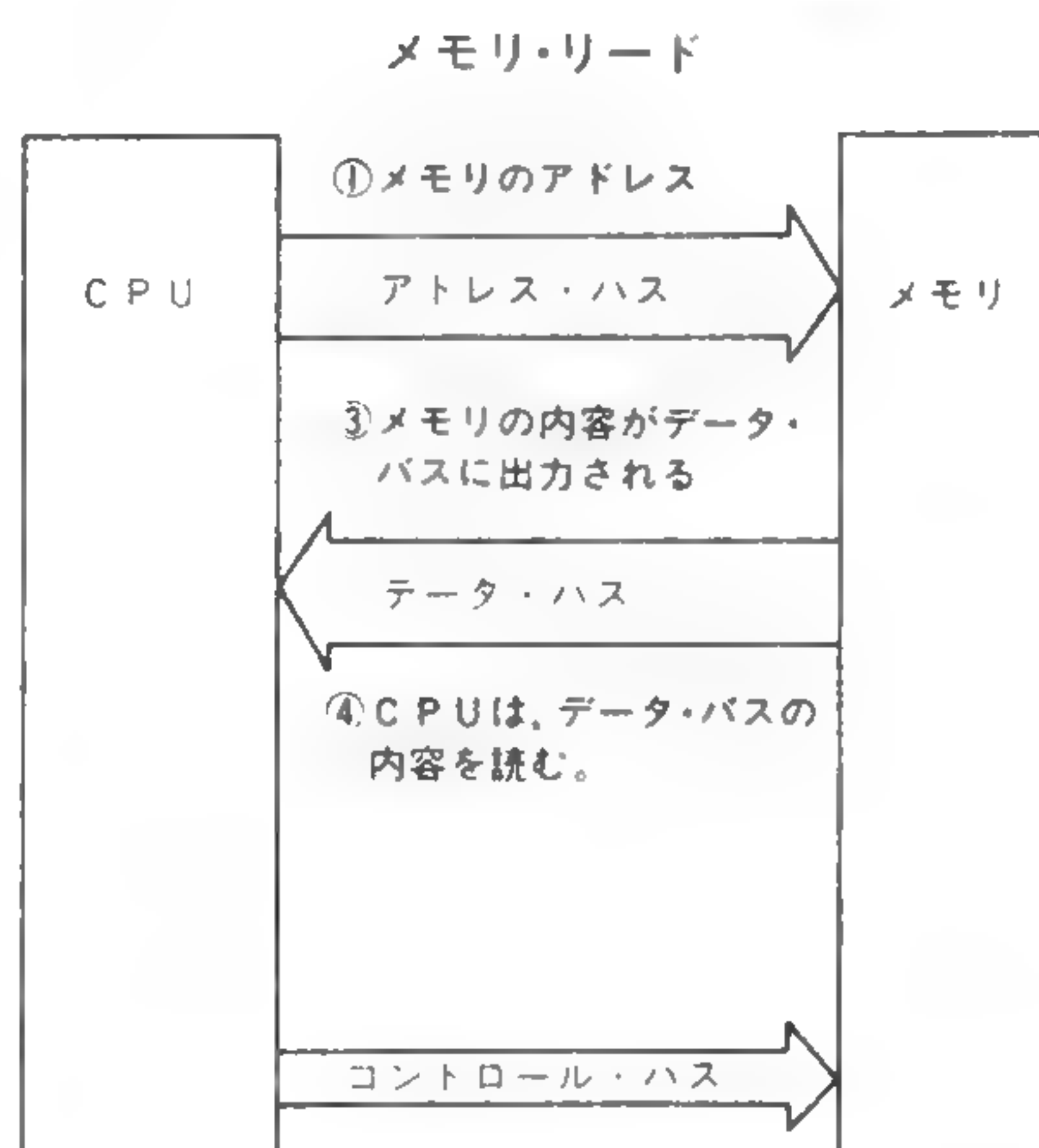
CPUがメモリのデータをリード/ライトする

とき、アドレス・バスを通じてCPUからメモリへリード/ライトしたいアドレスを指定します。同時にコントロール・バスを通じてリードするのかライトするのかも指定します。こうしてCPUはリード/ライトの準備を始めます。この一連の動作をアクセス (Access) といいます。

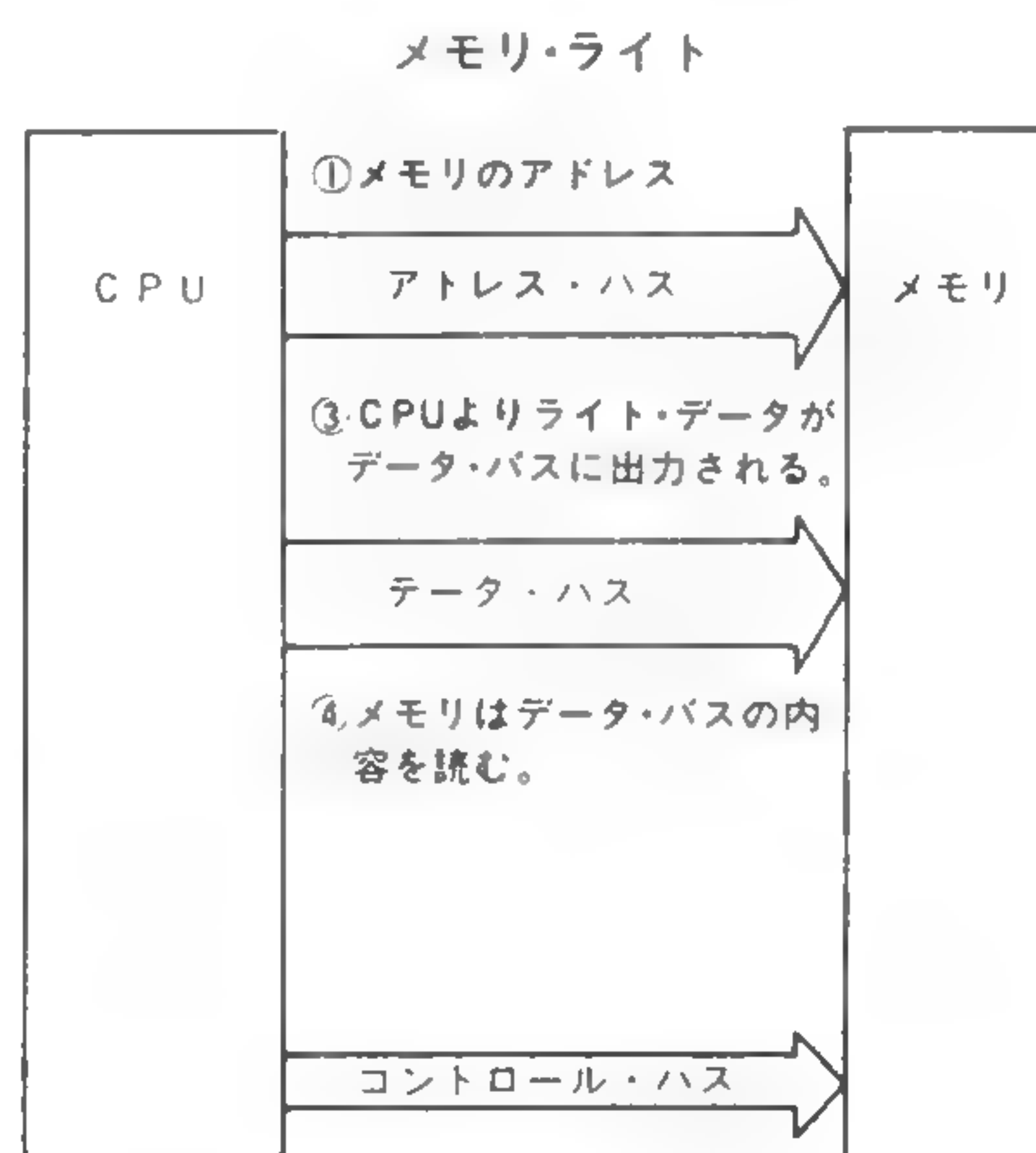
リードの場合、指定されたアドレスの内容がデータ・バスに出力されます。アドレスが指定されてからメモリの内容がデータ・バス上に出力されるまで数100n秒 (1n = 1/10億) の時間がかかります (この時間のことをアクセス・タイムといいます)。ライトの場合は、アドレスの指定に続きデータがCPUからバスを通してメモリに送られてきます。このときもメモリ内に正しく書き込むためにはやはり数100n秒の時間がかかります。こうしたリード/ライトのタイミングはすべてCPUにより制御されています。

I/Oポートに対するリード/ライトも同様の動作をしています。

図2-9 データのリード/ライト動作



- ②データ・バスの転送方向を「メモリ→CPU」に指定。
⑤データ・バスの転送方向を解除。



- ②データ・バスの転送方向を「CPU→メモリ」に指定。
⑤データ・バスの転送方向を解除。

マイコンで扱う数

1. 数の表現法

日頃私たちは、数を表わすのに数字を使っています。その中で最も使い慣れているのがいわゆる「10進数」です。10進数は0～9の数字を並べたものだということは誰もが知っています。ここで数とは物の個数や重さのような物理的な量を示すもので、数字とは数を識別するための記号と考え、数と数字とを区別します。たとえば、数「5」を表わす数字（記号）にも「5」、「五」、「V」、「Ⓜ」^③、……とさまざまな表現法があるのです。

マイコン（計算機）の世界では、2進数や16進数という表現法が使われています。2進数や16進数といった表現法は、見た目には私たちが日常使っている10進数とは異なりますが、表現規則は10進数と同じなのです。「5」という数を表現するのに前に挙げたようないくつかの方法があるのと同じことです。

数を表現するのには一定の規則があります。これは、誰が見てもその数字がある1つの値を示しているのだとわかるようにするためです。その規則の1番目は、数字（記号）をいくつ使って数を表現するかということです。たとえば、私たちが日常使っている数は、0～9までの10個の数字（記号）の組み合わせです。

2番目は、数の増えかたの規則です。私たち

がいつも使う数では1を1つ増すと2であって、突然5や6がくることはありません。つまり、数が1つ増すごとに右端の数字（記号）が次の数字（記号）に置きかわる、と決まっているわけです。

0	1	2	3	4	5	6	7	8	9
↑									↑
最低位									最高位

〈10進数の場合の数字の順序〉

3番目は、桁上がりの規則です。私たちがいつも使う数字の中では9が最高位だと決まっているので、2番目のきまりでいう「次の数字」がもうありません。このように、数を1つ増そうとしたときに右端の数字（記号）が最高位のものであったら、その数字を最低位の数字（記号）に置きかえ（10進数なら0）、1つ左（上の位）を次の数字に置きかえます。これが桁上がりです。もちろん桁上がりで生じた数字にも、同じ規則が適用されます。

以上3つの規則をまとめて計数体系と呼びます。これを私たちがいつも使っている数を例にして考えてみましょう。

まず、私たちは0～9の10種類の数字（記号）を使っていることが1番目の規則よりわかります。

2番目の規則により「1 2 3 4」を1つ増すと「1 2 3 5」となり、もう1つ増すと「1 2 3 6」となります。

③この数字（記号）の種類の個数のことをラディックス（radix＝基数）という。

④一般化したn進数では、nをラディックスとして考えるとえられる。ある数xの、n進数での表現は、

$$x = a_m \cdot n^m + a_{m-1} \cdot n^{m-1} + \cdots + a_1 \cdot n^1 + a_0 \cdot n^0$$

であるような $a_m, a_{m-1}, \dots, a_1, a_0$ を並べたものである。たとえば、 $n=10$ （10進数）の場合、 $924 = 9 \times 10^2 + 2 \times 10^1 + 4 \times 10^0$ となる。

次に3番目の規則により「1 2 3 9」を1つ増すと「1 2 4 0」となり、「1 2 9 9」を1つ増すと「1 3 0 0」,「9 9 9 9」では「1 0 0 0」となります。

この例のような計数体系のことを**10進法**といひ、10進法で表現されている数を**10進数**といひ⁶³のです。



2. 2進数と16進数

2進数(Binary number)とは、**2進法**(Binary notation)によって表わした数、つまり、数字を2種類使う計数体系のことです。数字としては「0」と「1」の2つを使います。0が最低位の数字で1が最高位の数字です。これは、前に**2章1-4 メモリ**のところ⁶⁴で述べたビットと対応します。ビットとはもともと「Binary digit」の略なのです。

第1章でも説明しました通り、コンピュータにとって数を表現する最小単位は、スイッチという「ON」「OFF」という2つの状態です。「ON」と「OFF」との中間の状態は存在せず、必ずどちらか一方の状態になります⁶⁵。コンピュータは「ON」と「OFF」、つまり「0」と「1」の2進数で記憶、演算、制御、入出力

をしています。

2進数も1桁では0と1の数しか表わせませんが、規則にそって桁数をふやすことによって任意の数を2進数で表わすことができるようになります。たとえば4桁の2進数では、10進数にして0~15の数を表わすことができます。

コンピュータの世界では、2進数のほかに8進数、16進数が使われます。8進数は昔はよく使われていましたが、現在では16進数の方が広く使われています。

数値などは、2進数より16進数をよく使います。なぜなら、2進数ではある程度大きな数になると桁が多くなって表示に場所をとるようになるし、だいいち、0と1ばかりが並んでいてはいくつなのかつかみにくいからです。16進数1桁は2進数4桁に相当しますので、桁数が2進数のときの1/4になります。また、使われる数字も0と1の2種類から16種類になるので見やすくなります。16進数で表現された数も、慣れてくればひと目見ただけでだいたいどのくらいかわかるようになってきます。

また、16進数が使われるもう1つの理由は、コンピュータの中で扱うデータが数だけではないということです。コンピュータでは、ビットの組み合わせで特定のパターン(模様)⁶⁶を表わすこともあります。このパターン・データを表わすのに2進数を使い、たとえば1のところは点があるというふうに決めておけば各ビットの状態が理解しやすいのですが、桁数が多くてあまり見やすいとはいえません。16進数を使うと、2進数の1と0の表現のようにはいきませんが、

⑥このような情報の表現法をデジタルという。これに対して、「中間の状態」が存在するような情報はアナログという。

⑦もっとも一般的なのは、画面に表示したりプリンタに印字したりする文字の形である。どの文字も点の集まりでできて

いて、多くの場合1つの点(ドット)を1ビットに対応させている。

比較的容易にパターンを知ることができます。
 このように、あるときは数値として、あるときはビットのパターンとして表わされる情報（メモリやレジスタの内容など）には、16進数による表現が適しています。

16進数 (Hexadecimal number) は、**16進法** (Hexadecimal notation) で表わした数、つまり16種類の数字を使う計数体系のことです。10～15は**A**から**F**のアルファベットで表現します。

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
										(10	11	12	13	14	15)
↑									↑						
最低位															最高位

表2-1は、10進数、2進数、16進数の対応を示したものです。

それではこれから、2進数→16進数変換、16進数→2進数変換、16進数→10進数変換、10進数→16進数変換のそれぞれの方法について例を挙げながら説明していきます。

1 2進数→16進数変換

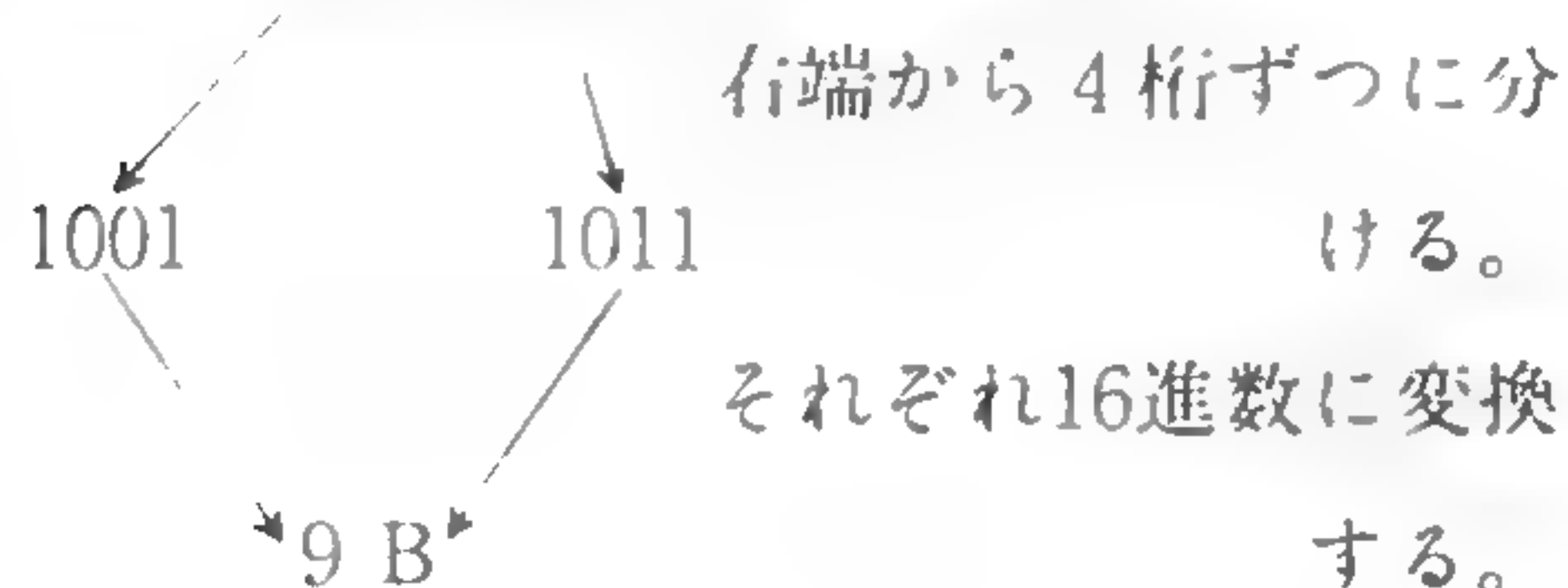
〈変換手順〉

- ① 2進数を右端から4桁ずつに分けます。
- ② 分けた4桁の2進数は16進数1桁と対応しますので、表2-1を参照しながら4桁ごとの2進数を16進数に変換します。

慣れてくれば16種の2進数と16進数との対応はすぐわかるようになってきます。そうなれば表を見なくても変換できるようになります。

〈例〉

2進数「**10011011**」を16進数に変換する。



これで2進数「**10011011**」が16進数「**9B**」に変換できました。

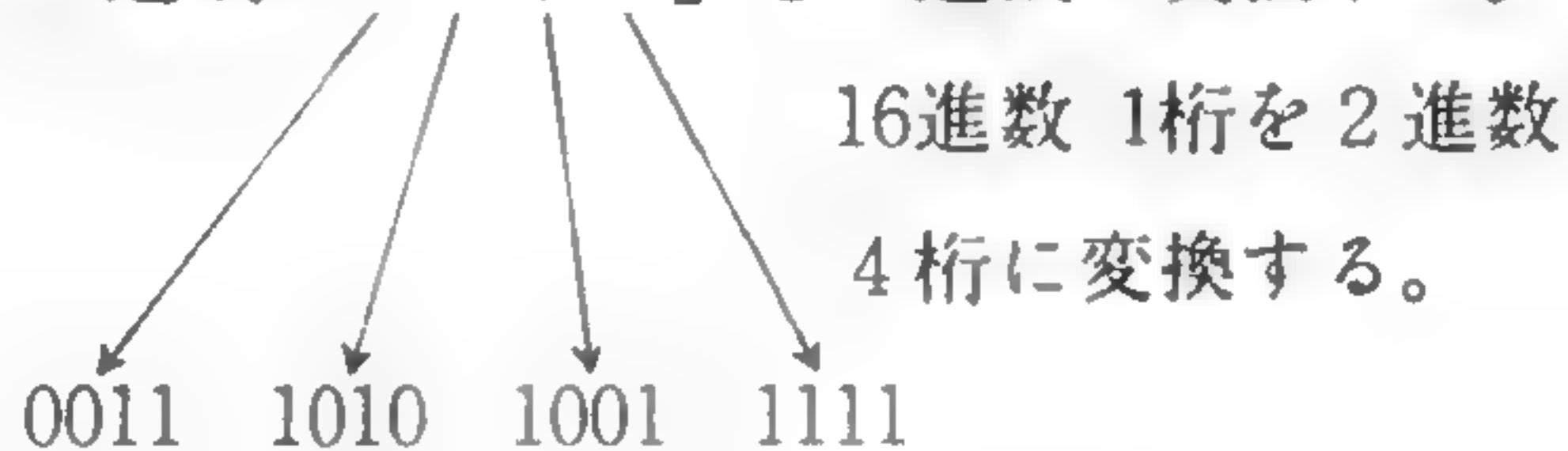
2 16進数→2進数変換

〈変換手順〉

16進数を1桁ずつ表2-1を参照しながら2進数4桁に変換していきます。これはまったく2進数→16進数変換の逆の作業です。

〈例〉

16進数「**3A9F**」を2進数に変換する。



これで16進数「**3A9F**」が2進数「**0011101010011111**」に変換できました。

表2-1 10進、2進、16進の対応

10進数	2進数	16進数
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
32	100000	20
64	1000000	40
127	1111111	7F
128	10000000	80
129	10000001	81
255	11111111	FF
256	100000000	100

3 16進数→10進数変換

〈変換手順〉

- ①16進数の各桁を10進数に変換します。
- ②次に①より求めた値を用いて次式を計算すれば10進数が求まります。

$$x = H_n \times 16^n + H_{n-1} \times 16^{n-1} + \dots + H_2 \times 16^2 + H_1 \times 16 + H_0$$

ただし、 x が求める10進数。

$H_n, H_{n-1}, \dots, H_1, H_0$ が16進数の各桁を10進数に変換した値

〈例〉

16進数(8C2F)を10進数に変換する。

$$\begin{aligned} x &= 8 \times 16^3 + 12 \times 16^2 + 2 \times 16 + 15 \\ &= 8 \times 4096 + 12 \times 256 + 2 \times 16 + 15 \\ &= 32768 + 3072 + 32 + 15 \\ &= 35887 \end{aligned}$$

これで16進数「8C2F」が10進数「35887」に変換できました。

4 10進数→16進数変換

〈変換手順〉

- ①10進数を16で割り、商とアマリを求めます。
- ②アマリは0～15までの数になるので、このアマリを16進数に変換します。
- ③商がゼロになるまで①～③を繰り返します。
- ④今まで求まった n 桁の16進数を初めに求めた16進数を右端の桁とし、最後に求めた16進数を左端として並べれば変換終わりです。

〈例〉

10進数(45000)を16進数に変換する。

$$\begin{aligned} 45000 \div 16 &= 2812 \dots 8 && 8 \text{ は16進数の } 8 \\ 2812 \div 16 &= 175 \dots 12 && 12 \text{ は16進数の } C \\ 175 \div 16 &= 10 \dots 15 && 15 \text{ は16進数の } F \\ 10 \div 16 &= 0 \dots 10 && 10 \text{ は16進数の } A \end{aligned}$$

AFC8

これで10進数「45000」が16進数「AFC8」に変換されました。

3. 負数と小数点以下の表現方法

これまでに説明した2進数や16進数には符号も小数点以下ありませんでしたが、実際のデータには負数や小数点以下を含む数もあります。今度はコンピュータの中でこれらのデータをどう扱っているか調べてみましょう。ただし、この内容は入門編の範囲を超えますので、難しければ飛ばしてもかまいません。

1 負数の表現

ふだん、私たちは負数を表現するのに「-」記号をつけます。コンピュータの中では、この符号も0と1で表わします。一般に「0」はプラス(+),「1」はマイナス(-)を示すのに使われます。

次頁の表2-2は、8ビット(1バイト)で表現できる符号つき整数です。参考までに、次頁の表2-3に8ビット(1バイト)で表現できる符号なし整数を載せておきました。

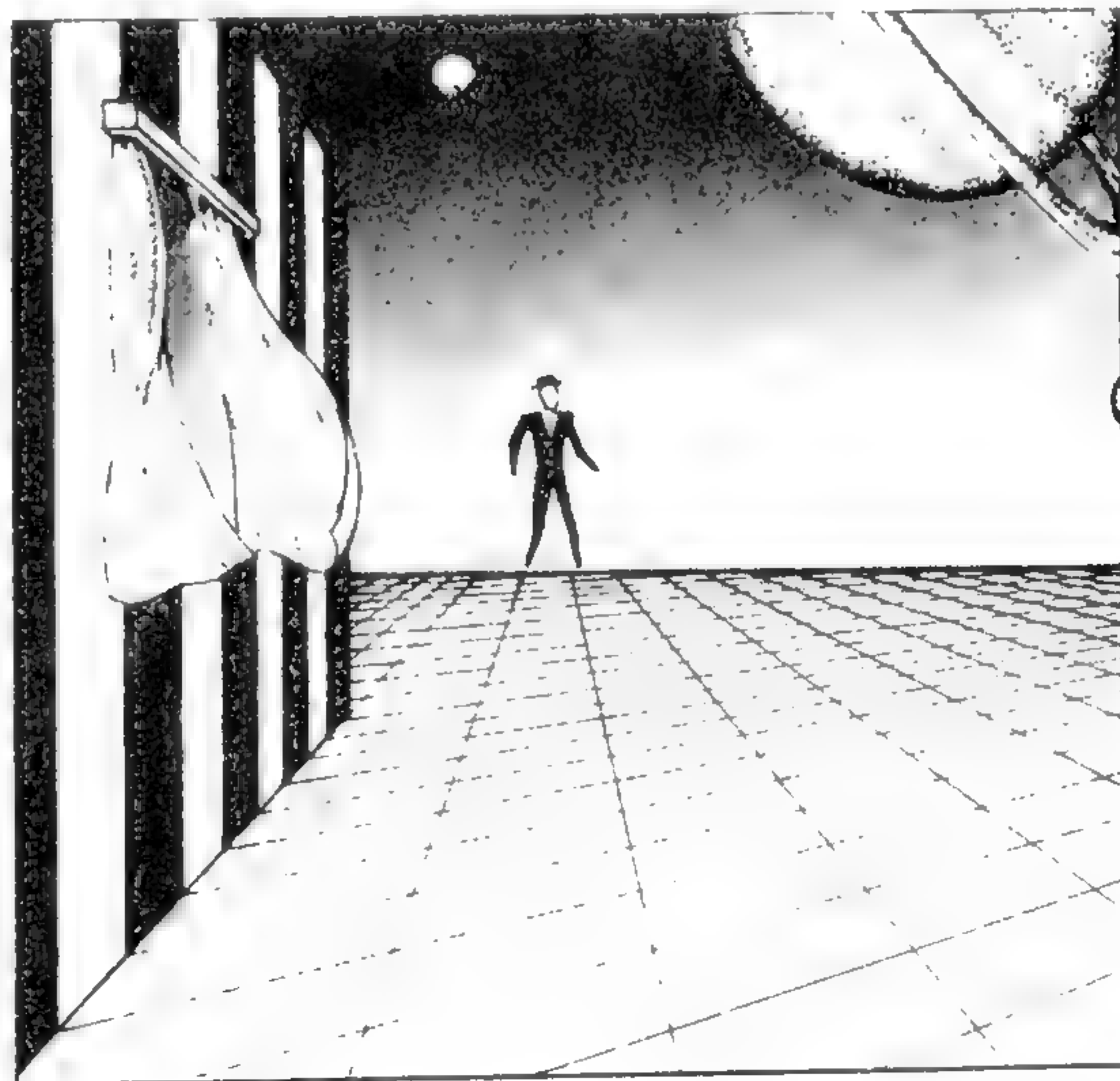


表2-2 2の補数表示による符号つき8ビットの値

10進数	2進数							
	符号	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
+127	0	1	1	1	1	1	1	1
+126	0	1	1	1	1	1	1	0
+125	0	1	1	1	1	1	0	1
⋮								
+64	0	1	0	0	0	0	0	0
⋮								
+16	0	0	0	1	0	0	0	0
⋮								
↑ プラス								
+3	0	0	0	0	0	0	1	1
+2	0	0	0	0	0	0	1	0
+1	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0
↓ マイナス								
-1	1	1	1	1	1	1	1	1
-2	1	1	1	1	1	1	1	0
-3	1	1	1	1	1	1	0	1
⋮								
-16	1	1	1	1	0	0	0	0
⋮								
-64	1	1	0	0	0	0	0	0
⋮								
-126	1	0	0	0	0	0	1	0
-127	1	0	0	0	0	0	0	1
-128	1	0	0	0	0	0	0	0

表2-3 符号なし8ビットの値

10進数	2進数							
	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
255	1	1	1	1	1	1	1	1
254	1	1	1	1	1	1	1	0
253	1	1	1	1	1	1	0	1
⋮								
129	1	0	0	0	0	0	0	1
128	1	0	0	0	0	0	0	0
127	0	1	1	1	1	1	1	1
⋮								
64	0	1	0	0	0	0	0	0
⋮								
16	0	0	0	1	0	0	0	0
⋮								
3	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0

⑯ 10進数で0から3を引けば0-3=-3になるように、2の補数で表現された負の2進数も、00000000(10進数で0)から00000011(10進数で3)を引くと11111101(10進数で-3)となる。

⑰ 2の補数に変換することは、10進数でいえば-1をかけることに相当する。つまり、10進数で+3×(-1)=-3、-3×(-1)=3を2進数で行なうのが2の補数変換である。

表2-2の2進数の最上位ビットが符号ビットです。この表を見ておわかりでしょうが、たとえば10進数の「+3」と「-3」は符号が違うだけで「3」という数字は変わりません。ところが2進数で+3を表わすと「00000011」、-3を表わすと「11111101」で、符号が変われば符号から右のビットも変わってしまっています。このような数の表現法を**2の補数表示**といいます。

8ビットの場合、符号なし整数ならば0~255(16進数でFF)まで表現できます。2の補数表示による整数なら-128(16進数で80)~0~+127(16進数で7F)まで表現できます。

ここで注意しなければならないのは、この例でいう符号なし整数の**255**と2の補数表示の**-1**がともに16進数では**FF**(2進数では11111111)であることです。これは、負数の-1~-128と符号なし整数の128~255についてもいえます。つまり、プログラムをつくる人がそのデータを符号なし整数として扱っているのか、符号つき整数として扱っているのかによって、同じ2進数でも異なった10進数値になってしまうのです。

2進数(16進数)で負数を表わすには、2の補数以外にも10進数と同じ考えかたで「符号プラス絶対値」とする方法もあります。どの方法を使うかは、プログラムをつくる人がそのプログラムの特性によって決めればよいでしょう。

● 2の補数への変換方法^{⑱)}

<変換手順>

- ① 変換しようとする2進数のビットを反転します。つまり「0」→「1」、「1」→「0」

2の補数変換の代わりに-1をかけても同じことではあるが、2の補数変換より時間がかかる。

⑱ この操作のことを1の補数変換という。

㉑ ここでいう「重み」とは、桁が上がる(下がる)につれて同じ数字(たとえば2)が20になったり20000になったりすることを指す。つまり、10進数で「1020」の「2」は「20」

にします。⁽³⁸⁾

②反転した値に1を加えると2の補数になります(2進の加算については2章3 2進演算を参照)。

〈例〉

①8ビットの2進「00000011」(10進数の3)を2の補数にする。

00000011 (10進数の3)
↓ 全ビットを反転する
11111100
↓ +1する
11111101 (10進数の-3)

②いま2の補数にした2進数を、再度2の補数に変換してみる。

11111101
↓ 全ビットを反転する
00000010
↓ +1する
00000011 (もとの数に戻る)

2 小数点以下の数の表現

2進数(16進数)による小数点以下の数も、整数と同様に扱うことができます。すなわち小数点を基準にして小数点以下1桁目は 2^{-1} という重みを持っています。これは、10進数の 10^{-1} という⁽³⁹⁾ような重みの考え方に相当します。

たとえば10進数の2.6875は、

$$\begin{aligned} 2.6875 &= 2 + 0.5 + 0.125 + 0.0625 \\ &= 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \\ &\quad \times 2^{-3} + 1 \times 2^{-4} \end{aligned}$$

(2×10^1)の重みを持つし、「21540」の「2」は「20000」(2×10^4)の重みを持つ。

④⑩0.1(10進数) = 0.0625 + 0.03125 + 0.00390625 + 0.00195312
(2^{-4}) (2^{-5}) (2^{-8}) (2^{-9})

5 + ……

④⑪小数点以下が24ビットなら $2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13}$

$$= 10.1011 \text{ (2進数)}$$

となります。

ここで注意しなければならないのは、10進数から2進数(またはその逆)の変換をするときに小数点以上の数(整数部)は正確に行なうことができるのに、小数点以下の数(小数部)には誤差が出てしまうことです。

たとえば、10進数の0.1を2進数に変換すると、

$$0.0001100110011 \dots$$

と「0011」が繰り返されます。⁽⁴⁰⁾

逆にこの値を10進数に変換すると、小数点以下の桁が24ビットもあれば10進数にしたとき0.099999… \div 0.1となるのに、⁽⁴¹⁾小数点以下の桁が5ビットしかないとなってしまう。

小数点以下の数を扱う場合には、このようなことがあるので十分注意が必要です。

4. その他の表現法

ここでは、これまで説明したもの以外でよく使われる数の表現法BCDと浮動小数点について簡単に説明します。難しいと思われれば、ここも飛ばしてもかまいません。

1 BCD (Binary Coded Decimal : 2進化10進)

これまで説明してきた数は2進数でしたが、コンピュータの中では10進数も扱うことができます。MSXのZ80A CPUにも10進数の演算のための命令があります。

$+2^{-16} + 2^{-17} + 2^{-20} + 2^{-21} + 2^{-24} + 2^{-25} + 2^{-28} + 2^{-29} + 2^{-32} + 2^{-33}$ まで表わせるから。5ビットだと $2^{-4} + 2^{-5} = 0.0625 + 0.03125 = 0.09375$ となる。ちなみに 2^{-33} とは $1/2^{33} \div 1/86億 (\div 0.0000000011641532)$ のことである。

コンピュータの中で、10進数はBCDという表現法で表わします。BCDとは、10進数の「0～9」を、それに対応する4ビット2進数の「0000～1001」で置きかえて10進数を表わす方法です。

たとえば、10進数の「139」はBCDで「0001 0011 1001」と表わします。

BCDは、みかけは2進数ですが基本的には10進数なので、人間にとってたいへんわかりやすいのです。しかし、2進数に比べて計算に時間がかかるし、同じ数を表わすのに2進数よりビットを多く使います。

② 浮動小数点 (Floating point)

浮動小数点とは、小数点の位置を一定にしないで、指数というもので小数を表現するものです。たとえば、123.45を 0.12345×10^3 というふうに表わせれば、 10^3 の3のところを変えるだけで1.2345にも12.345にもなりますから、0.12345のところは変えなくてすむという、非常に便利な表現法です。この0.12345のところを仮数部、 10^3 の3のところを指数部といいます。

浮動小数点は10進数（BCDも含む）では、次の形で表わされます。

$$m \times 10^e$$

m：仮数部 (Mantissa)

10^e ：指数部 (Exponent)

仮数部は、小数点以下の数字で表わすと決められています。⁽⁴²⁾

たとえば、仮数部が0.32918、指数部が 10^3 なら、

$$0.32918 \times 10^3 = 329.18$$

を表わします。

浮動小数点を2進数で扱う場合は、次の形で表現します。

$$(-1)^s \times m \times 2^e \quad \text{または} \quad (-1)^s \times m \times 16^e$$

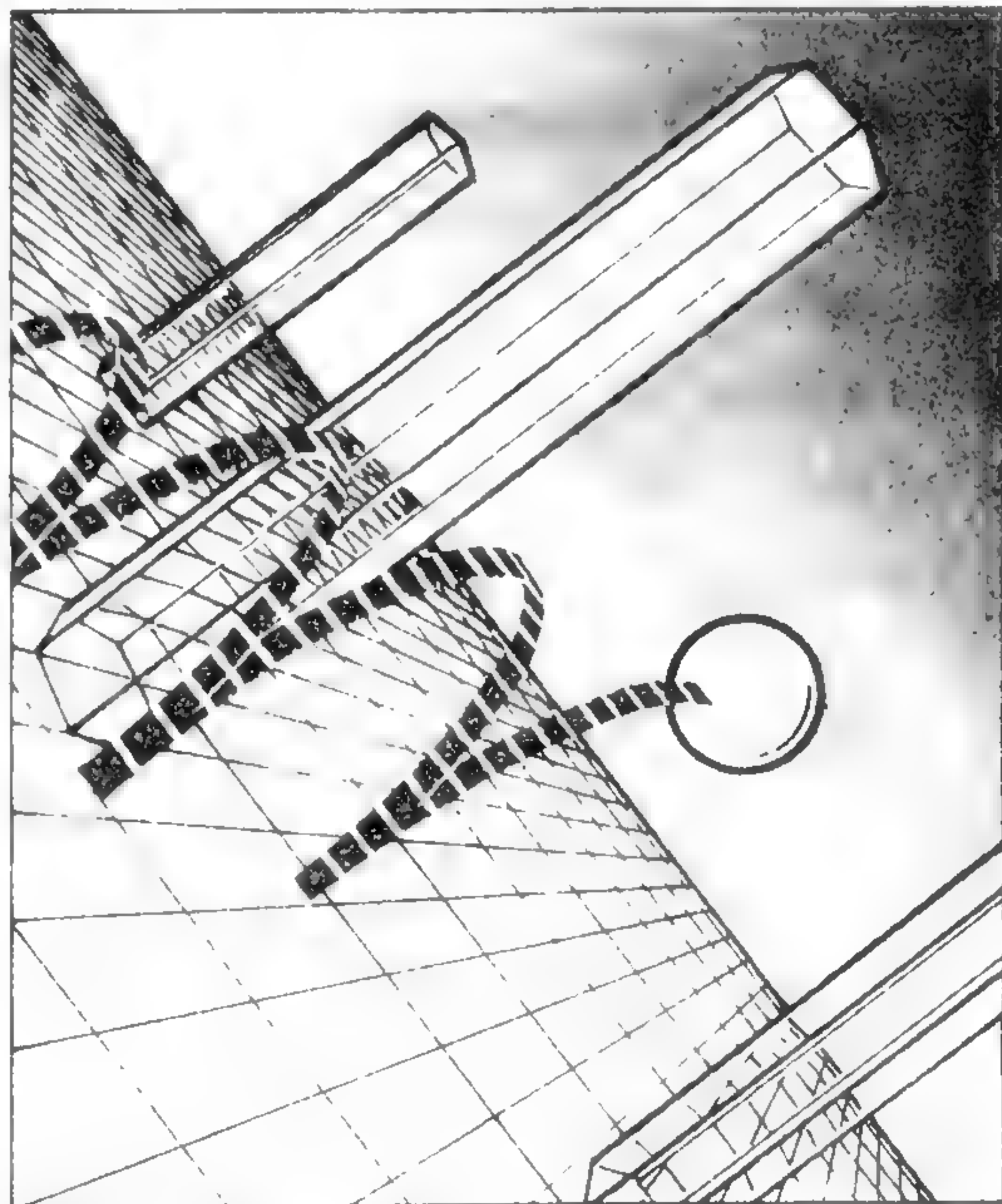
s：仮数部の符号 (0：プラス, 1：マイナス)

m：仮数部

2^e ：指数部 (2^e の場合は仮数部を1ビット単位で正規化し、 16^e の場合は4ビット単位で正規化します)

2進数での浮動小数点の演算は、演算回数が増すにつれて演算結果の誤差が大きくなるという欠点がありますが、10進数による浮動小数点演算より速く計算できます。

マシン語のプログラムで、浮動小数点を使って計算することはあまりありません。ほとんどの場合2進数の整数による計算ですみます。いかえれば浮動小数点を使うような複雑な計算はBASICなどの高級言語でプログラムし、マシン語ではなるべく整数で表わせる数を使うようにするということです。



④ 仮数を、このように定められた範囲の数に調節することを正規化 (Normalize) という

④ たとえば、5をかける計算は同じものを5回足せばよい。また、5で割った答えは、何回「5」が引けるかということである。



2進演算

この節では、コンピュータが2進数を使ってどのように演算（広い意味での計算）しているかについて説明します。演算には算術演算（いわゆる四則計算）、論理演算、シフト、ローテイトがあります。

1. 算術演算 (Arithmetic operation)

算術演算とはいわゆる四則計算ですから、加算・減算・乗算・除算の4つの計算を指します。しかしMSXのZ80ACPUには乗算・除算の命令がなく、それらは加減算の組み合わせで表現します⁽⁴⁾。ここでは、加算と減算について述べることにします。

1 加算

初めに1ビット（2進数ひとけた）どうしの加算について考えてみましょう。

1ビットどうしの加算には次の4通りの組み合わせがあります。

0	0	1	1
+ 0	+ 1	+ 0	+ 1
0	1	1	1 0
			↑
			桁上がりが発生 (キャリー)

これを見ておわかりのとおり、1ビットの2進数の加算では、

- ① 同一の数（0と0、1と1）の加算では、その桁の結果は0になり、異なる数（0と1、1と0）の加算では、その桁の結果は1になる。

- ② 1と1の加算では、桁上がり（キャリー：Carry）が発生する。

ということがわかります。

それでは、次に複数ビットの加算ではどうでしょう。この場合は、1ビットずつ右端の桁から計算していけばよいのです。ただし、計算途中でキャリーが発生したら、次の桁ではそのキャリーも含めて計算しなければいけません。

たとえば、2つの8ビットの符号なし2進数「01110101」と「01010100」の加算をしてみましょう。

例題) 01110101 (10進数の117)
+ 01010100 (10進数の84)

 ?

- ① 01110101 右端の桁（ビット0）から計
+ 01010100 算を始めます。

 1 1 + 0 = 1でキャリーなし。
- ② 01110101 次の桁（ビット1）を計算。
+ 01010100 0 + 0 = 0でキャリーなし。

 01
- ③ 01110101 次の桁（ビット2）を計算。
+ 01010100 1 + 1 = 10で、その桁は0、

 001 キャリー発生。
- 1 ←キャリー
- ④ 01110101 次の桁（ビット3）はビット2か
+ 01010100 らのキャリー（1）も含めて計算。

 1001 1 + 0 + 0 = 1でキャリーなし。

- ⑤ 01110101 次の桁（ビット4）を計算。

$$\begin{array}{r} 01110101 \\ +01010100 \\ \hline 01001 \end{array}$$
 $1 + 1 = 10$ で、その桁は0、キ
 ャリー発生。

1 ← キャリー

- ⑥ 01110101 次の桁（ビット5）はビット4

$$\begin{array}{r} 01110101 \\ +01010100 \\ \hline 001001 \end{array}$$
 からのキャリー（1）も含めて
 計算。 $1 + 1 + 0 = 10$ でその桁
 は0、キャリー発生。

1 ← キャリー

- ⑦ 01110101 次の桁（ビット6）はビット5

$$\begin{array}{r} 01110101 \\ +01010100 \\ \hline 1001001 \end{array}$$
 からのキャリー（1）も含めて
 計算。 $1 + 1 + 1 = 11$ でその桁
 は1、キャリー発生。

1 ← キャリー

- ⑧ 01110101 次の桁（ビット7）はビット6

$$\begin{array}{r} 01110101 \\ +01010100 \\ \hline 11001001 \end{array}$$
 からのキャリー（1）も含めて
 計算。 $1 + 0 + 0 = 1$ でその桁
 は1、キャリーなし。

以上、加算結果は「11001001」（10進数の201）

となります。

MSXのZ80ACPUは、基本的には8ビット（1バイト）単位で計算します。計算結果は、キャリー フラグ⁽⁴⁾1ビットと値8ビットで求められます。たとえば、 $11111111 + 11111111$ の結果は、キャリー フラグ=1、値=11111110となります。

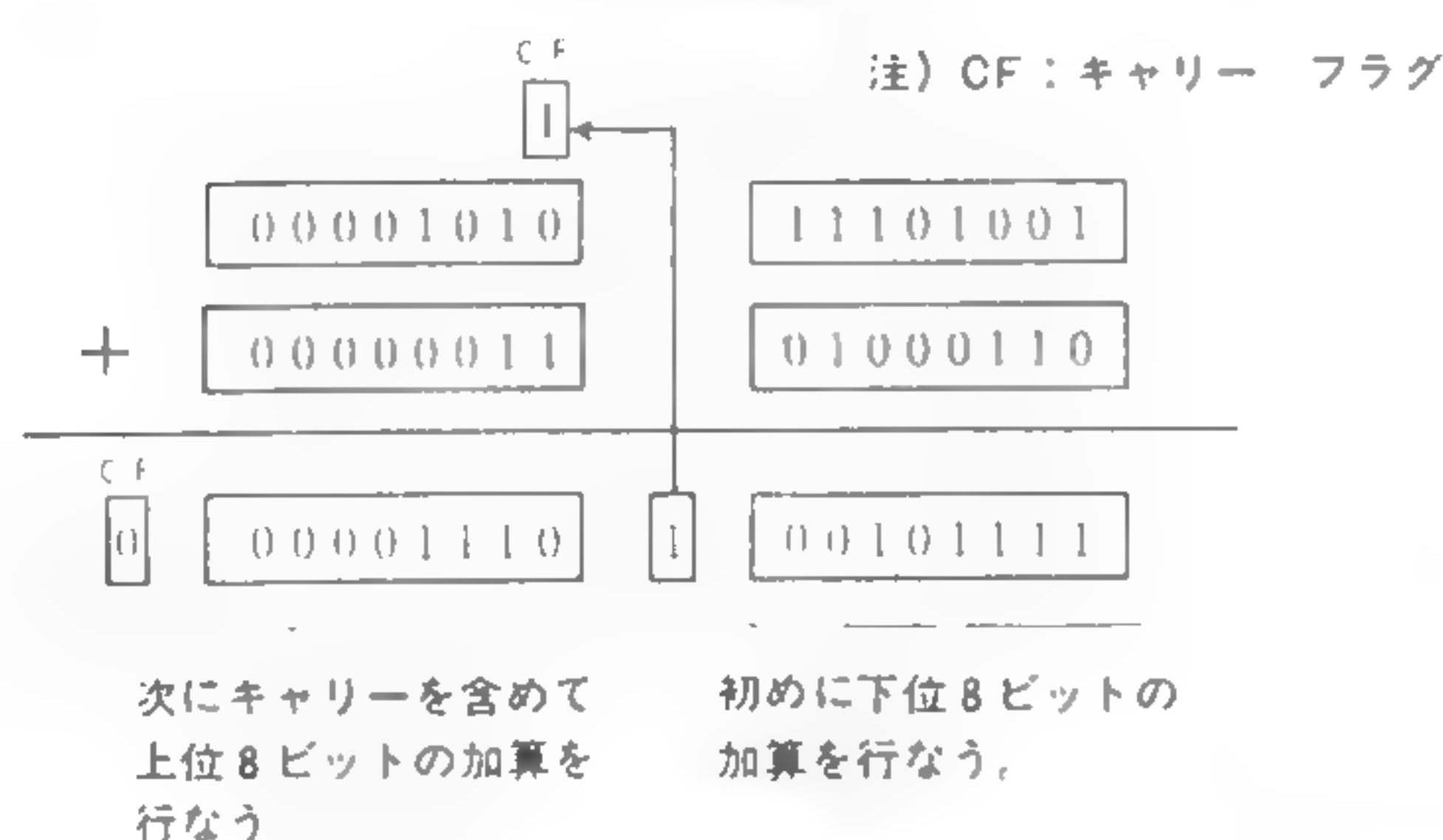
8ビットでは、符号なし整数で0～255までしか表わせません。キャリー フラグを使うことで256以上の数でも計算することができるようになります。たとえば1つのデータが2バイトで表わされている場合は、図2-10のように計算することで結果が求められます。

表2-4に、バイト数と表わせる数との対応を載せておきました。

表2-4 バイト数と表わせる数

バイト数	表わせる数	
	符号なし整数	符号付き整数
1	0～255	-128 ～ +127
2	0～65535	-32768 ～ +32767
3	0～16777215	-8388608 ～ +8388607
4	0～4294967295	-2147483648 ～ +2147483647

図2-10 2バイト(16ビット)の加算



$$\begin{array}{r} 0000101011101001 \text{ (10進数の2793)} \\ +0000001101000110 \text{ (10進数の838)} \\ \hline 0000111000101111 \text{ (10進数の3631)} \end{array}$$

④ キャリー フラグとは、計算の結果キャリーが発生したことを示す印のこと。キャリーがあったかどうか常時見張っている見張り番が、キャリーが発生した途端に手に持った旗(=フラグ)を上げて教えてくれると思えばよい。フラグはキャ

リーのほかにも数種類あって、みなそれぞれの役割で見張っている。フラグが集めてあるのが「フラグ・レジスタ（Fレジスタ）」である。詳しくは3章1.1(1)(35ページ)参照。

また、ここでのキャリー フラグは、8ビットからの桁あ

2 減算

加算のときと同じく、初めに1ビットの2進数どうしの減算について考えてみましょう。

1ビットどうしの減算には次の4通りの組み合わせがあります。

$$\begin{array}{r} 0 \\ -0 \\ \hline 0 \end{array} \quad \begin{array}{r} 1 \\ -1 \\ \hline 0 \end{array} \quad \begin{array}{r} 1 \\ -0 \\ \hline 1 \end{array} \quad \begin{array}{r} 0 \\ -1 \\ \hline ? \end{array}$$

↓

上のビットから1を
借⁽⁴⁵⁾りてきて計算。
(ボローが発生)

$$\begin{array}{r} 10 \\ -1 \\ \hline 1 \end{array}$$

1ビットの2進数の減算では、

- ① 引く数と引かれる数が同一の数(0と0, 1と1)の場合は、その桁の結果は0となり、引く数と引かれる数が異なる数(0と1, 1と0)の場合は、その桁の結果は1になる。

- ② 引く数が1, 引かれる数が0の場合借⁽⁴⁵⁾り(ボロー: Borrow)が発生する。

という規則があります。

それでは、次に複数ビットの減算についてみましょう。やはりこの場合も、加算のときのように右端から1ビットずつ減算していきます。計算途中でボローが発生したら、発生した次の桁でさらに-1(ボローの分)します。

たとえば、8ビットの符号なし2進数「01000101」から同じく8ビットの符号なし2進数「00110011」を減算してみましょう。

例題)

$$\begin{array}{r} 01000101 \text{ (10進数の69)} \\ -00110011 \text{ (10進数の51)} \\ \hline ? \end{array}$$

- ① 01000101 右端の桁(ビット0)から計算
-00110011 を始めます。
0 1-1=0, ボローはなし。

- ② 01000101 次の桁(ビット1)を計算。
-00110011 0-1=1, ボロー発生。
10

- ③ 01000101 次の桁(ビット2)はボローも含めて計算。
- 1←ボロー
010 (1-0)-1=0, ボローなし。

- ④ 01000101 次の桁(ビット3)を計算。
-00110011 0-0=0, ボローなし。
0010

- ⑤ 01000101 次の桁(ビット4)を計算。
-00110011 0-1=1, ボロー発生。
10010

- ⑥ 01000101 次の桁(ビット5)はボローも含めて計算。
- 1←ボロー
010010 (0-1)-1=0, ボロー発生。⁽⁴⁶⁾

- ⑦ 01000101 次の桁(ビット6)はボローも含めて計算。
- 1←ボロー
0010010 (1-0)-1=0, ボローなし。

ふれの有無を示している。

④ (通常の10進数の筆算のように)上位の桁から「1」を借りてきて計算すること。

⑥③の(1-0)-1=0でボローが発生しなかったのに、⑥

の(0-1)-1=0でボローが発生しているのは、カッコの中を先に計算しているからである。つまり、カッコの中でボローが発生したかどうか判定している。カッコの中は元からある数字で、カッコの外の「1」は下位桁のボローである。

⑧ 01000101 次の桁（ビット7）を計算。

-00110011 0 - 0 = 0, ボローなし。

00010010

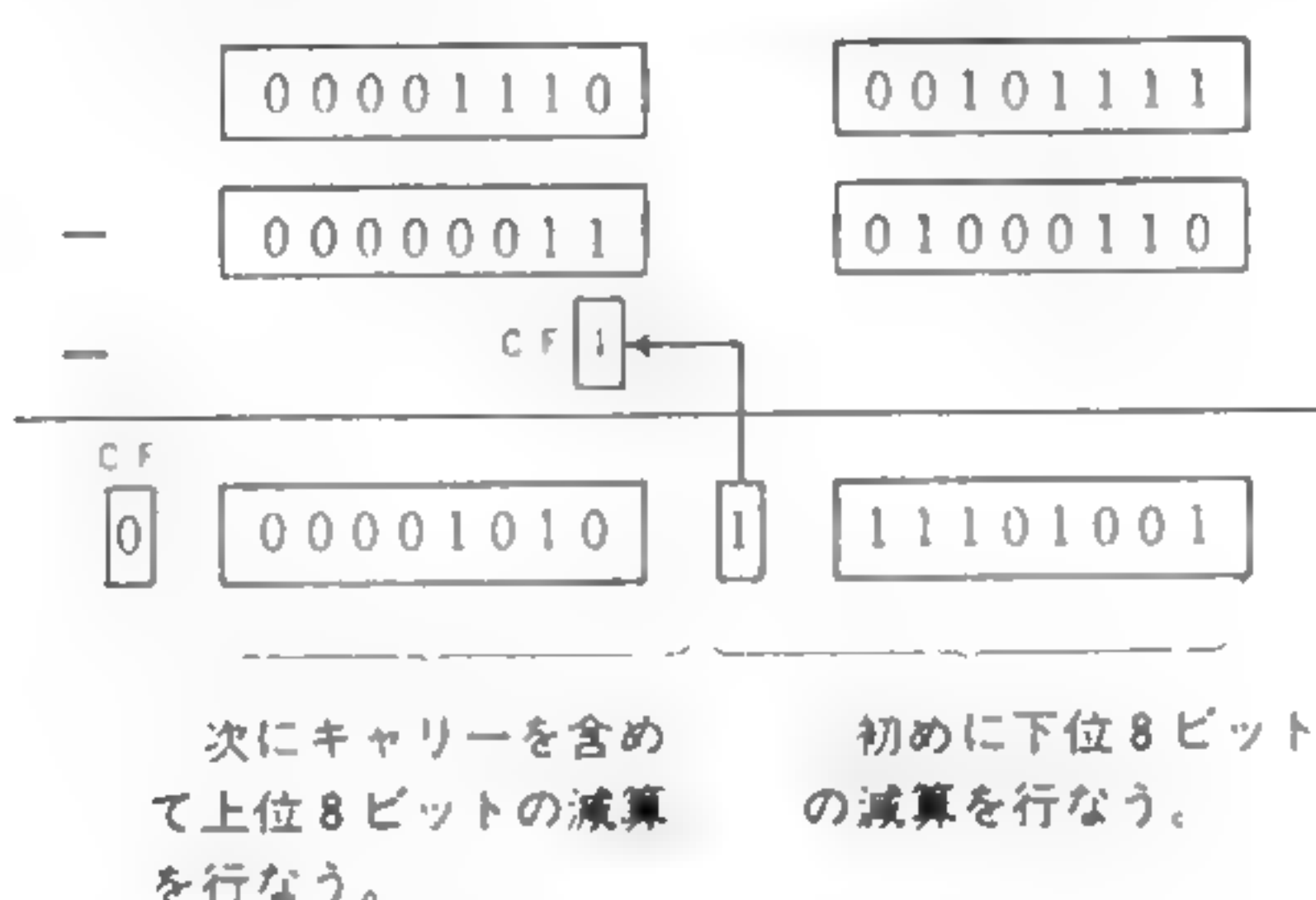
以上、減算結果は「00010010」（10進数の18）となります。

MSXのZ80A CPUには、ボロー フラグ

（ボローがあったことを示す印）というものは
ありません。キャリー フラグがボロー フラグ
を兼用しています。キャリー フラグを使えば、
数バイトにわたる数値でも減算ができます。図
2-11に、例として1つのデータが2バイトで表
わされる値どうしの減算を示します。

図2-11 2バイト(16ビット)の減算

注) CF: キャリー フラグ



0000111000101111 (10進数の3631)
- 0000001101000110 (10進数の838)

0000101011101001 (10進数の2793)

3 2の補数表示と加減算

いままでの説明は、符号なし整数についての
加減算でしたが、今度は符号付き整数(2の補数表
示による)の加減算について考えてみましょう。

たとえば、-13（8ビットの2進数で「1111
0011」）に20（8ビットの2進数で「00010100」
）を加算してみると、

11110011
+00010100

00000111 (10進数の7)

と、-13+20の計算結果として7が求められます。

次に、3（8ビットの2進数で「00000011」）
から-5（8ビットの2進数で「11111011」）
を減算してみると、

00000011
-11111011

00001000 (10進数の8)

と、3-(-5)の計算結果として8が求められます。

このように、負数を2の補数で表わしておけ

ば、符号なし2進数と同様に演算できます。

正負混合計算であっても、数が正か負かを意
識せずに計算できるのが2の補数という表現法
なのです。

図2-12に8ビットの符号付き整数の計算例
をいくつか示しておきます。

図2-12 8ビットの2の補数の計算例

①	+29	00011101
	+)-96	10100000
	-----	10111101
②	-100	10011100
	+)-80	01010000
	-----	11101100
③	+100	01100100
	+)-100	10011100
	-----	00000000
④	-8	11111000
	-)-2	11111110
	-----	11111010
⑤	0	00000000
	-)-1	00000001
	-----	11111111
⑥	-100	10011100
	-)-10	00001010
	-----	10010010

2. 論理演算 (Logic operation)

論理演算とは、ブール代数の演算をするためのものです。ブール代数は、小学校や中学校で勉強した「集合」が基本になっています。集合やブール代数については、別の参考書を読んでもらうこととし、ここでは論理演算について述べることに⁽⁴⁷⁾しましょう。

論理演算で扱う値には「真 (True)」と「偽 (False)」の2つしかありません。これは、2進数の1ビットの値「1」と「0」に対応するものです。コンピュータの中では「真」は「1」、「偽」は「0」として演算します。

論理演算の基本的な演算は、**NOT** (否定)、**AND** (論理積)、**OR** (論理和) の3種類です。そのほかに、**XOR** (排他的論理和)、**IMP** (包含)、**EQV** (同値) などの演算もあります。⁽⁴⁸⁾ **XOR**、**IMP**、**EQV** の論理計算は、**NOT**、**AND**、**OR** を組み合わせてつくることができます。⁽⁴⁹⁾

図2-13にこれらの真理値表 (Truth table, 真理表とも呼ぶ) を示します。また、これらの論理計算を図にしたもの (ベン図表: Ven Diagram) を図2-14に示します⁽⁵⁰⁾

MSXのZ80A CPUでは、8ビット単位で論理演算を行ないます。これを示したのが図2-15です。Z80Aは、**NOT**、**AND**、**OR**、**XOR** の4種類の論理演算命令を備えています。

図2-13 各論理演算の真理値表

X	Y	NOT X	X AND Y	X OR Y	X XOR Y	X IMP Y	X EQV Y
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	0	0
1	1	0	1	1	0	1	1

図2-14 各論理演算を図で表わす

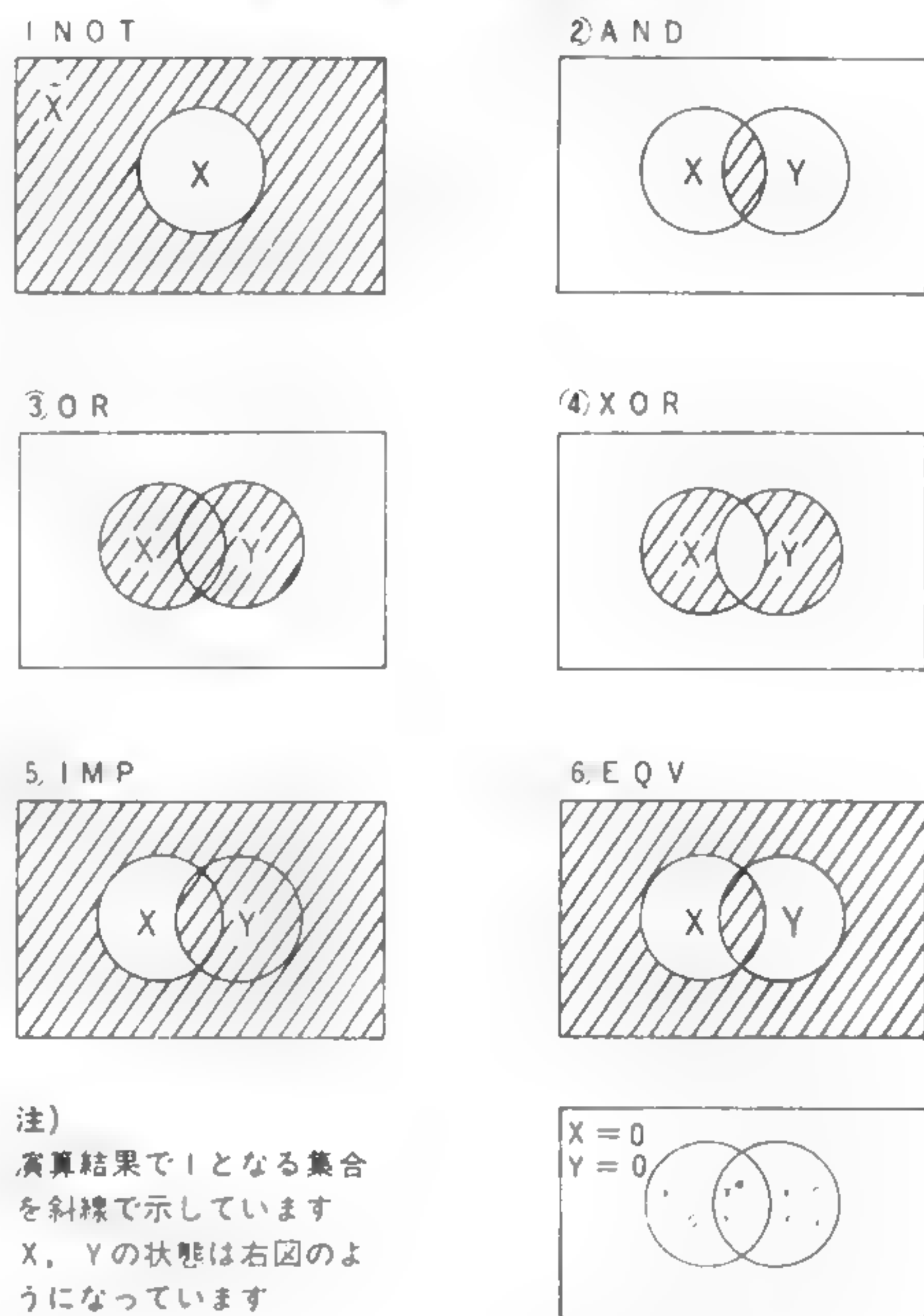


図2-15 1バイト(8ビット)の論理演算

NOT)	1 0 1 0 1 0 1 0	AND)	1 0 1 0 1 0 1 0
	0 1 0 1 0 1 0 1		0 0 1 1 0 0 1 1
	0 1 0 1 0 1 0 1		0 0 1 0 0 0 1 0
XOR)	0 0 1 1 0 0 1 1	OR)	1 0 1 0 1 0 1 0
	0 1 1 0 0 1 1 0		0 0 1 1 0 0 1 1
			1 0 1 1 1 0 1 1

④ 論理演算は、コンピュータに使われているデジタル回路 (AND回路, OR回路など) と同じ動作をCPUの命令でするための演算である。つまり、デジタル信号の0と1だけでできた情報を使ってする計算である。たとえば、1と1とが出会ったとき、結果を1にするか0にするかは論理演算の種類によって決まる。

⑩ **NOT** (否定) は、入力が「0」なら「1」、「1」なら「0」を出力する演算。**AND** (論理積) は、入力される数値がすべて「1」なら「1」を出力、1つでも「0」があれ

ば0を出力する演算。**OR** (論理和) は、入力される数値のうちで1つでも「1」があれば「1」を出力、すべて「0」なら「0」を出力する演算である。

⑪ たとえば **XOR** (排他的論理和) は、入力をA, Bとすると、 $((\text{NOT } A) \text{ AND } B) \text{ OR } (A \text{ AND } (\text{NOT } B))$ と表わすことができる。

⑫ 集合の包含関係を表示するのによく使われる、円を用いた図法。

論理演算を使って複数のビットをセット（1にする）したりリセット（0にする）したり、あるいは反転（0→1，1→0）させたりすることができます。

任意のビットをセット（1）するにはORを，リセット（0）するにはANDを使います。また，任意のビットを反転させるのにはXORが使われます（図2-16，17，18）。

図2-16 ORによるビット・セット

① 2進数「01100100」のビット7，6，3を1にする。

```

      0 1 1 0 0 1 0 0
OR)  1 1 1 0 0 1 0 0
-----
      1 1 1 0 0 1 0 0
    
```

.....ビット7，6，3を1にし，
のこりを0にする。

.....ビット7,3が0から1になった

.....ビット6は，もともと1だったので変わらない。

② 2進数「00000110」（10進数の6）を文字の“6”（ASCIIコードで「00110110」）にする。

```

      0 0 0 0 0 1 1 0
OR)  0 0 1 1 0 1 1 0
-----
      0 0 1 1 0 1 1 0
    
```

.....ASCIIコードの“0”の文字

.....ASCIIコードの“6”の文字
なった

図2-17 ANDによるビット・リセット

① 2進数「01100100」のビット6，2，1を0にする。

```

      0 1 1 0 0 1 0 0
AND) 1 0 1 1 1 0 0 1
-----
      0 0 1 0 0 0 0 0
    
```

.....ビット6,2,1を0にし，のこりを1にする。

.....ビット1はもともと0だったので変わらない

.....ビット6,2が1から0になった

② 文字“9”（ASCIIコード「00111001」）を2進数「00001001」（10進数の9）にする。

```

      0 0 1 1 1 0 0 1
AND) 0 0 0 0 1 1 1 1
-----
      0 0 0 0 1 0 0 1
    
```

.....下位4ビットのみ抽出する。

.....2進数の「00001001」（10進数の9）になる。

図2-18 XORによるビットの反転

① 2進数「01010110」の上位4ビットを反転する。

```

      0 1 0 1 0 1 1 0
XOR) 1 1 1 1 1 0 0 0
-----
      1 0 1 0 0 1 1 0
    
```

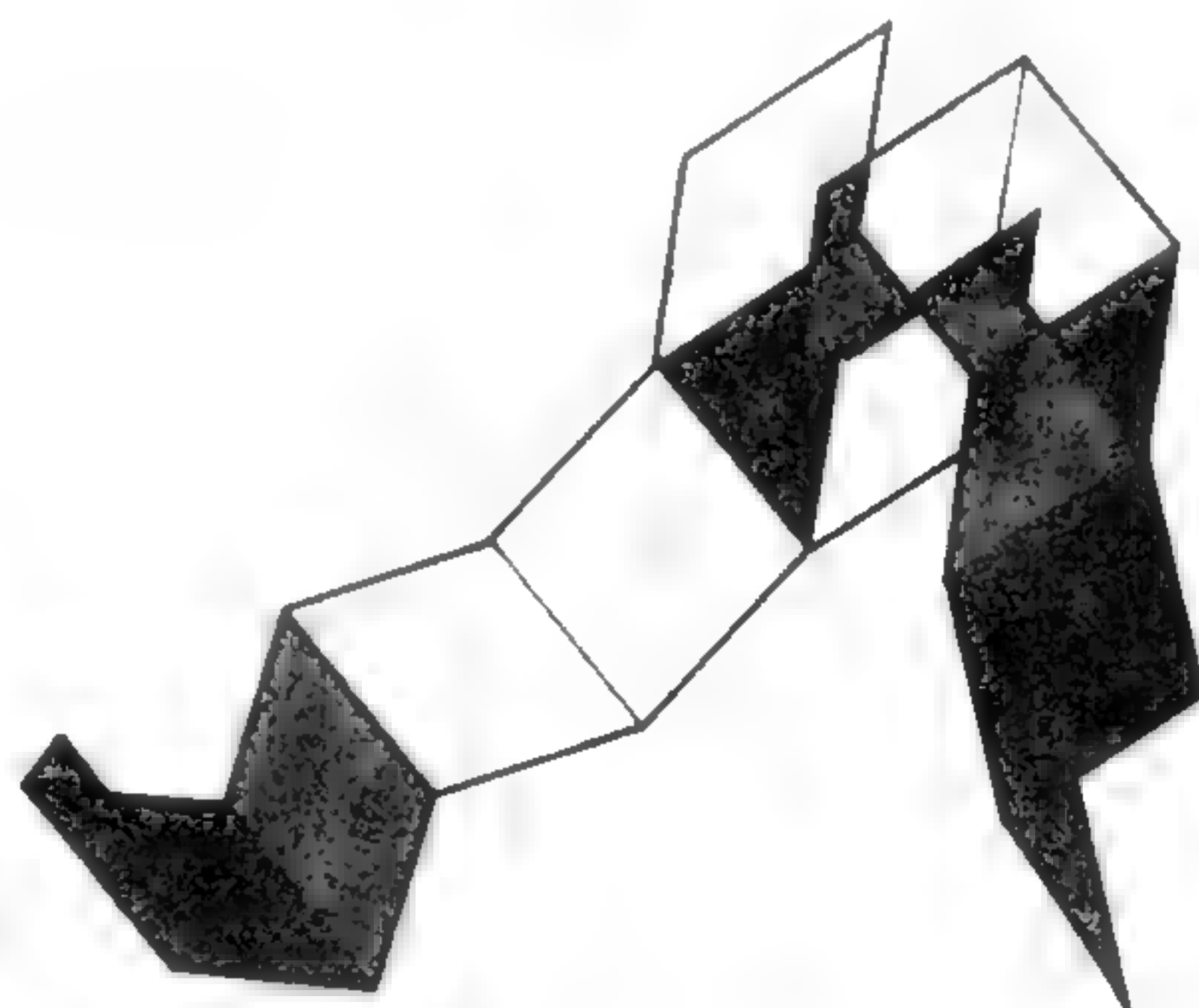
.....反転したいビットを1にし，
のこりを0にする
上位4ビットが反転した

② 2進数「11110000」のビット7，5，3，1を反転する

```

      1 1 1 1 0 0 0 0
XOR) 1 0 1 0 1 0 1 0
-----
      0 1 0 1 0 0 1 0
    
```

.....反転したいビット7,5,3,1を1
にし，のこりを0にする
.....ビット7,5,3,1が反転した。



3. シフトとローテイト

コンピュータの中では、加算や減算の算術演算や、NOT、AND、ORの論理演算のほかに、データの内容を左右に移動（シフト）したり回転（ローテイト）したりする命令があります。それがシフトとローテイトです。MSXのZ80A CPUは、1命令で（8ビットをひとかたまりとして）1ビットずつシフト/ローテイトさせることができます。

1 シフト

シフトには右へ移動するものと左へ移動するものがあり、演算と同じように論理シフトと算術シフトとがあります。

① 論理シフト (Logic shift)

論理シフトは、ビットを右/左に移動させて空いたところに0を入れ、はみ出した値をキャリー フラグに入れます。これを図にしたのが図2-19と図2-20です。

② 算術シフト (Arithmetic shift)

算術シフトとは、シフトによって符号ビット(第7ビット)が変化しないシフトのことです。

算術左シフトは本来、図2-21のように動作するシフト⁵¹⁾で、2の補数で表わされたある値xを左へ1ビットシフトします。これは $x \times 2$ の計算をしたのと同じ意味になります。

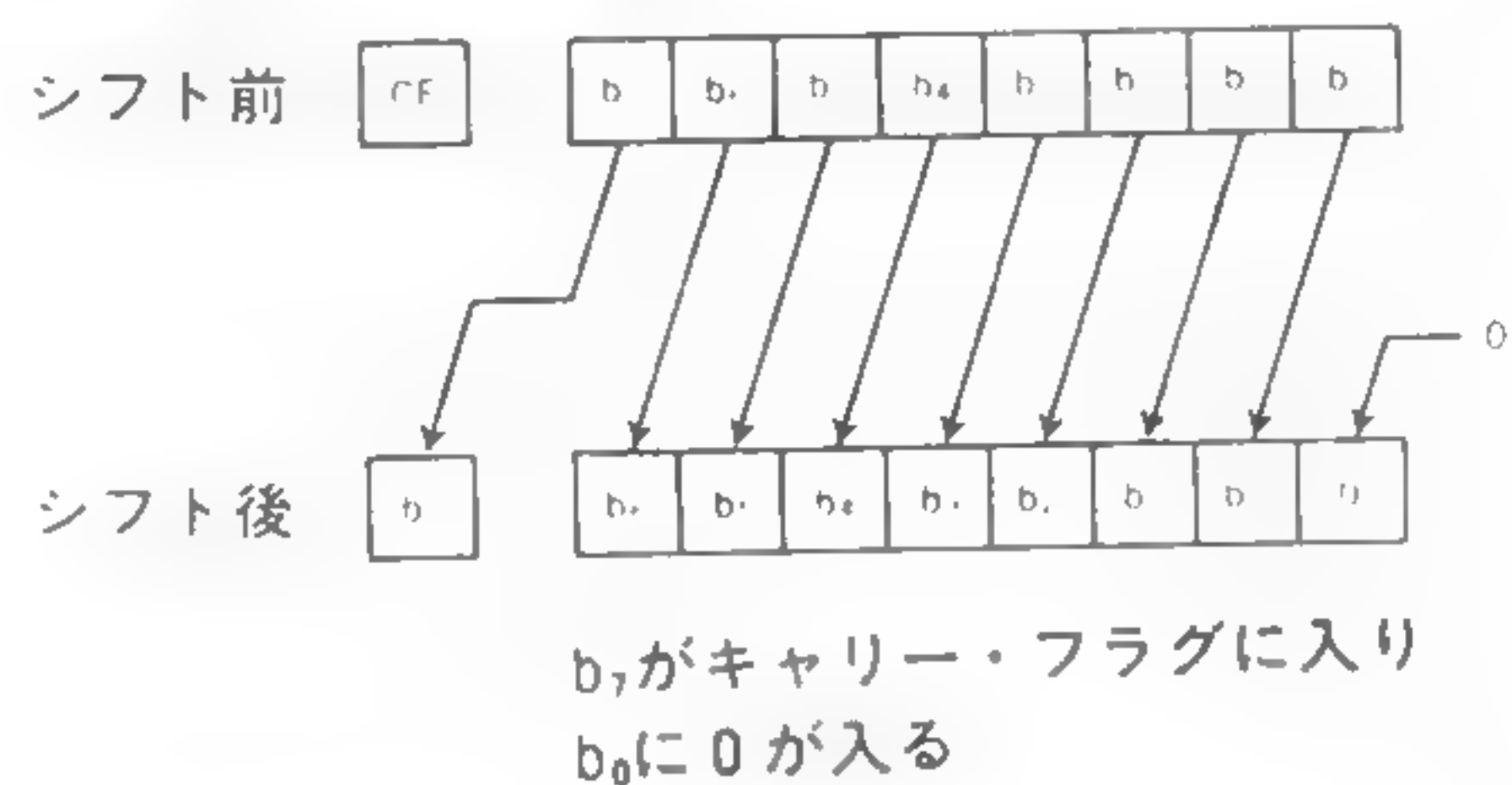
算術右シフトは、図2-22のようにシフト後も符号ビット（図では b_7 ）が変化しません。

算術右シフトは、2の補数で表わされた値

xを右へ1ビットシフトするもので、 $x \div 2$ を計算したのと同じ意味になります。⁵²⁾

図2-19 論理左シフト

● 論理左シフトの動作



● 論理左シフトの例

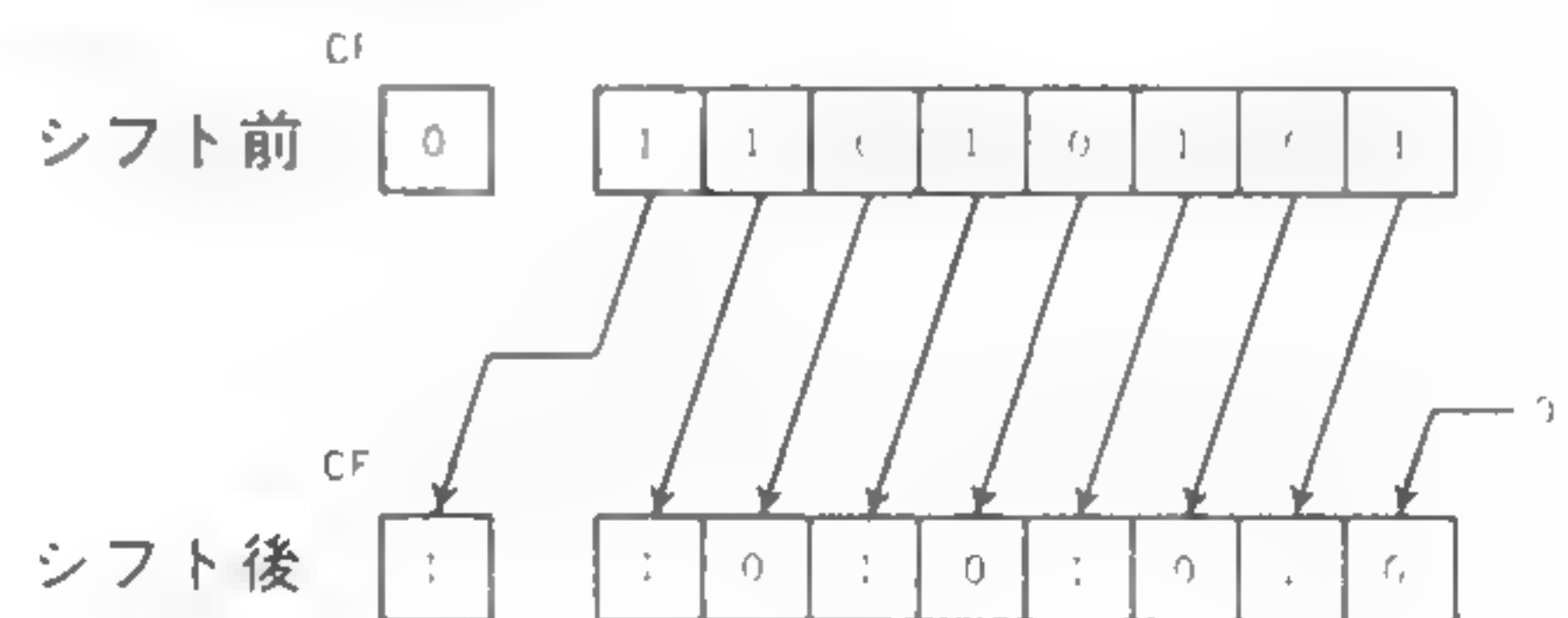
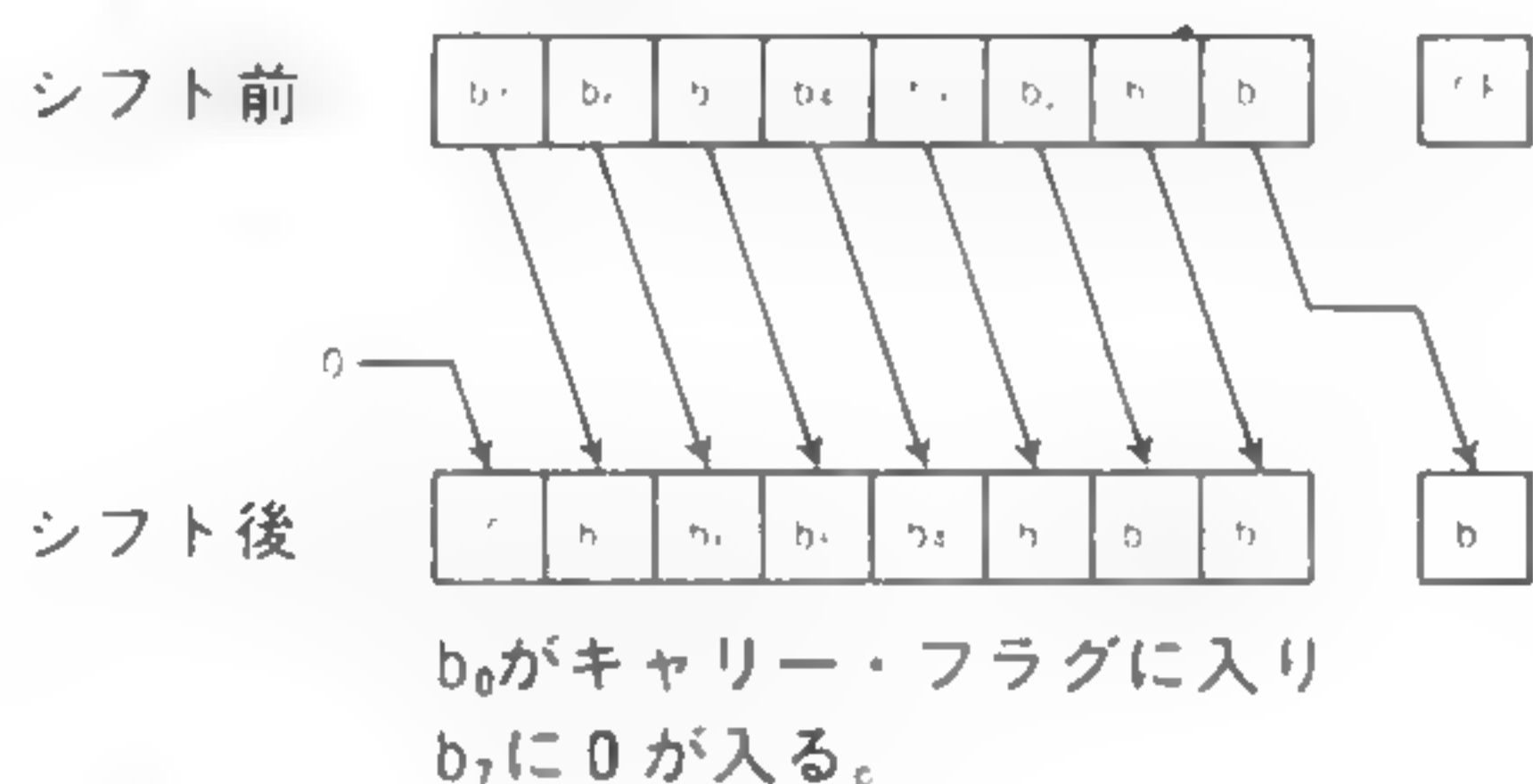
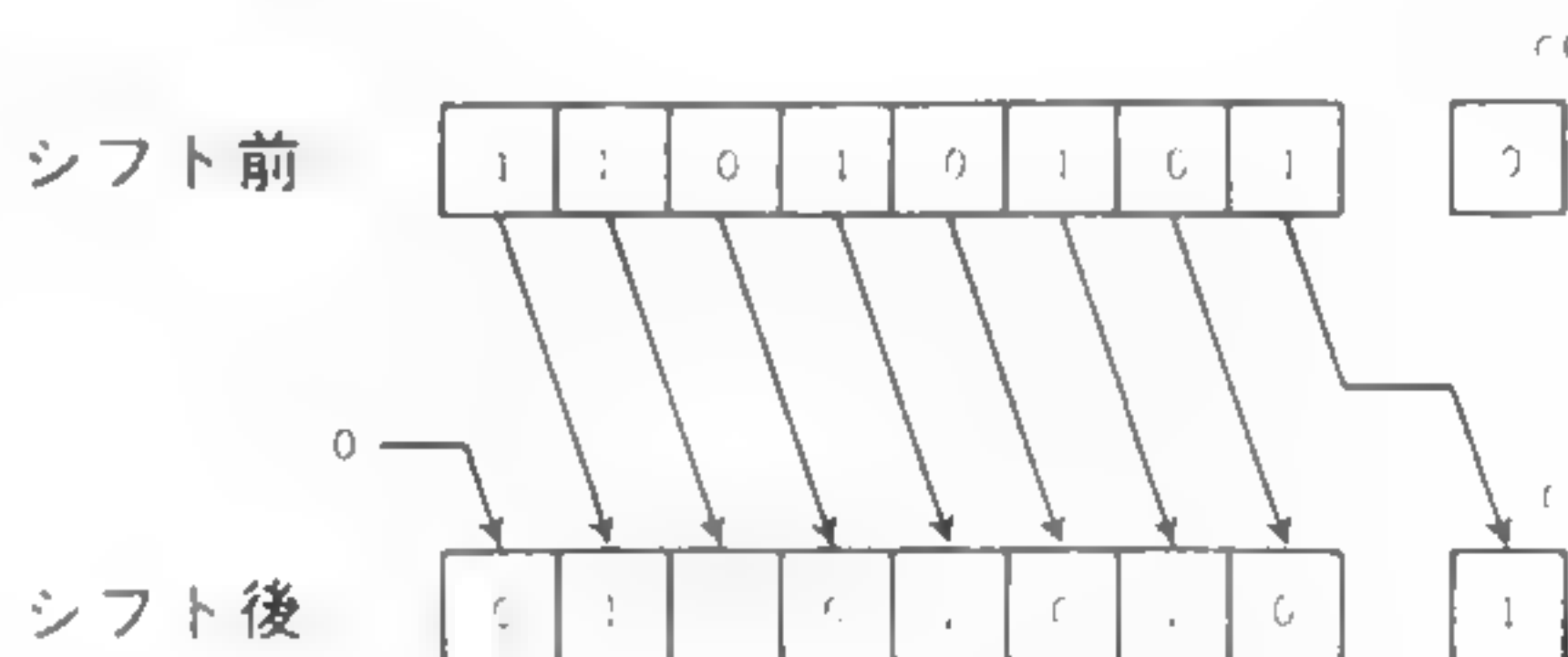


図2-20 論理右シフト

● 論理右シフトの動作



● 論理右シフトの例



⑤1 「データの内容」とは、メモリやレジスタの中のデータのビット構成のことである。移動（シフト）は単にビットをとなりのビットへ移すだけ（あふれた分は別途処理される）なのに対し、回転（ローテイト）は8ビットを環状につなげてビットを移動させる。

⑤2 なぜかマイコン、パソコンの分野では、算術左シフトと論

理左シフトを同じものとして扱っていたり、論理シフトのことを算術シフトと呼んでいたりする。

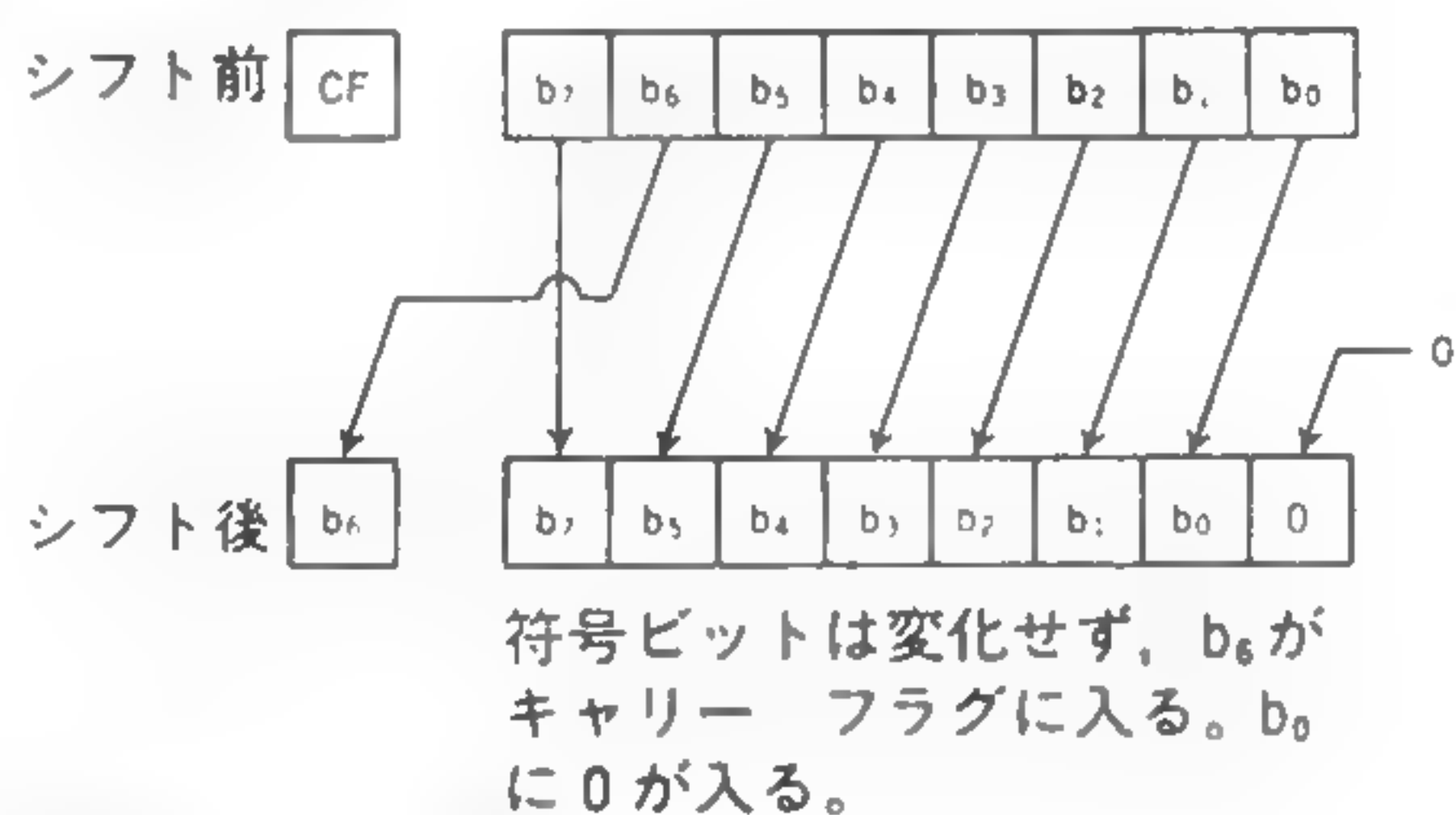
⑤3 ただし、計算結果はその値の小数部分を切り捨てた整数で求められる。これをBASICの式で表わすと、

$RESULT = INT(x/2)$

となる。よって、 $x = -1$ のときの計算結果は-1になる。

図 2-21 本来の算術左シフト

●算術左シフトの動作



●算術左シフトの例

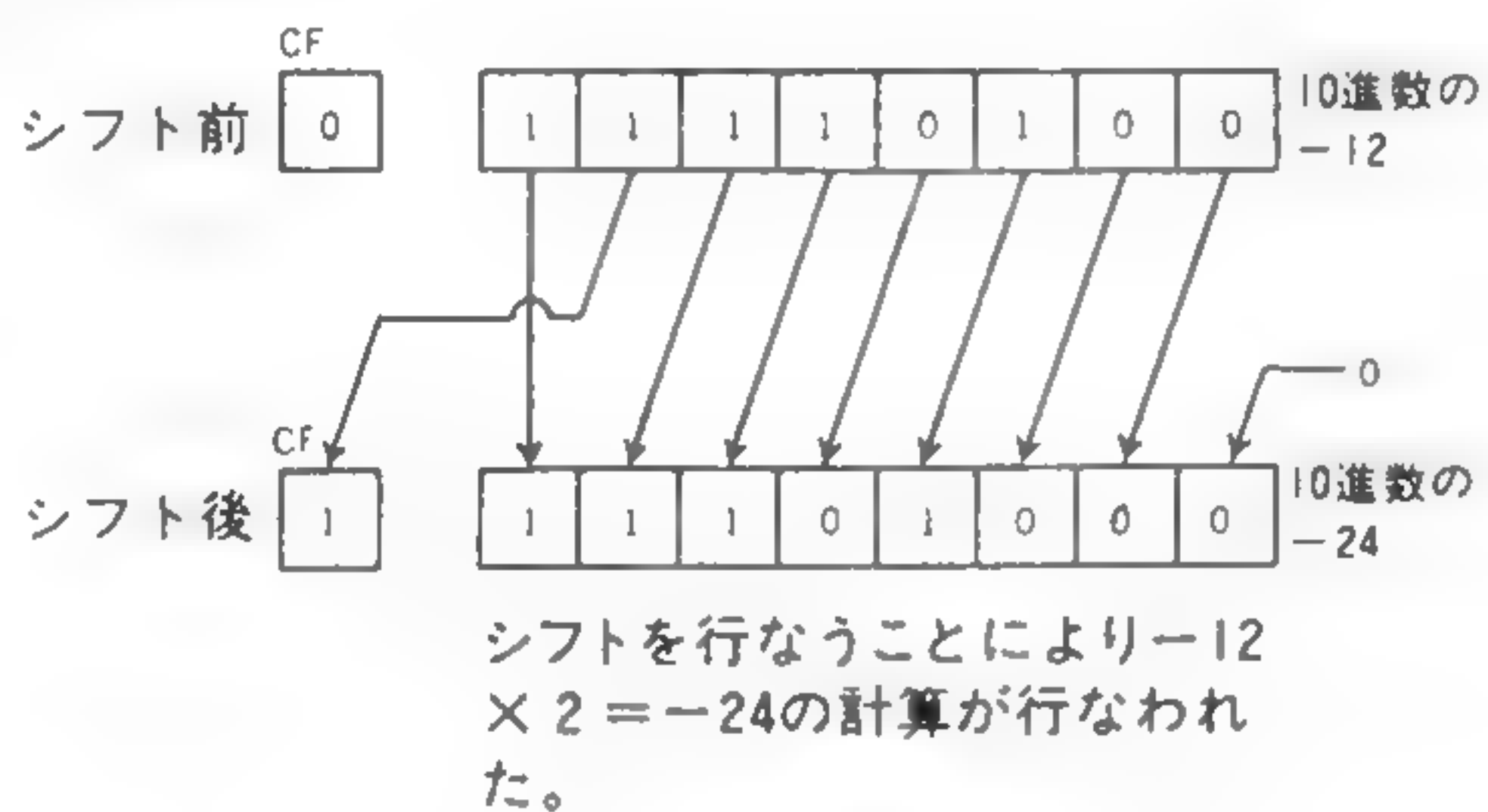
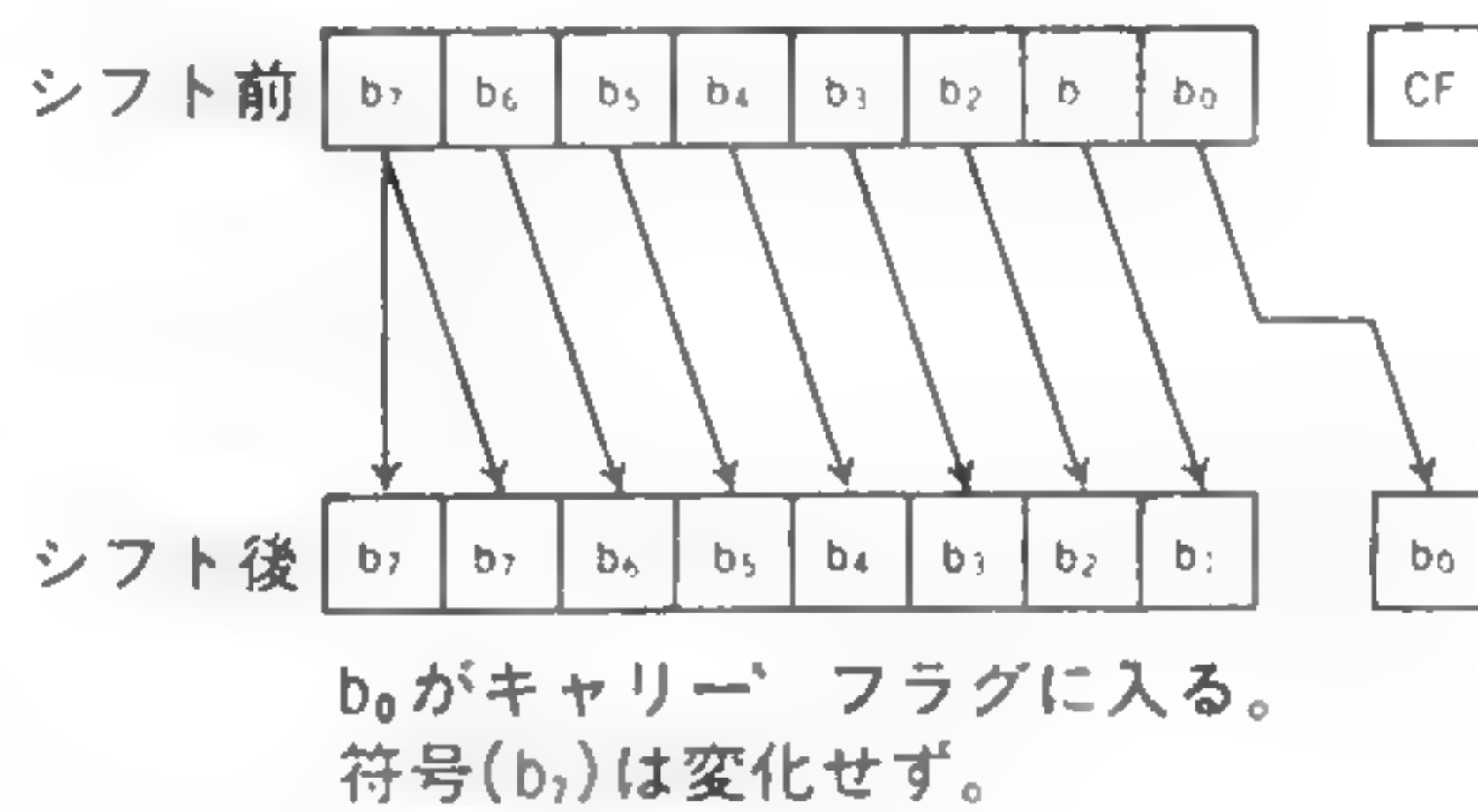
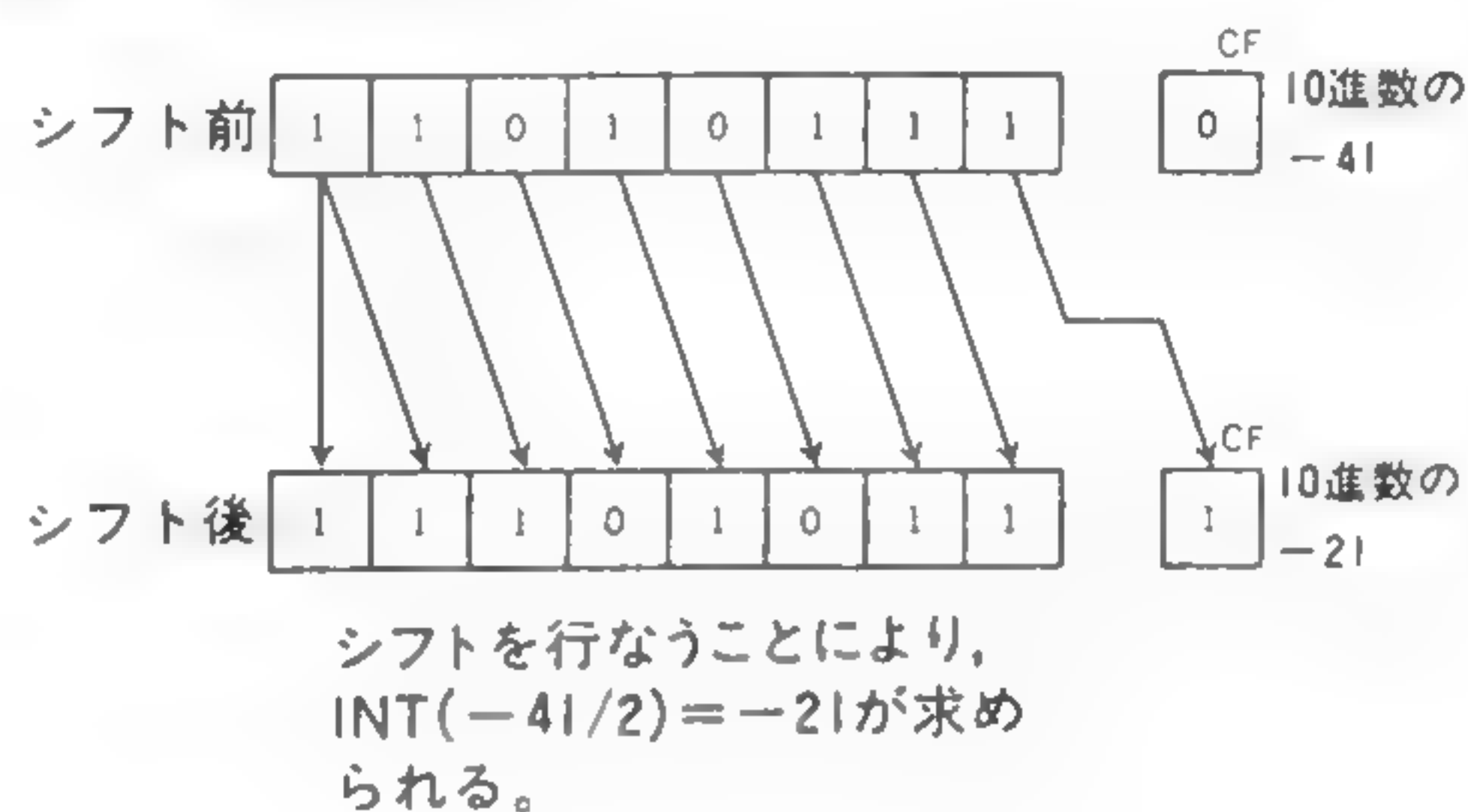


図 2-22 算術右シフト

●算術右シフトの動作



●算術右シフトの例



ローテイト

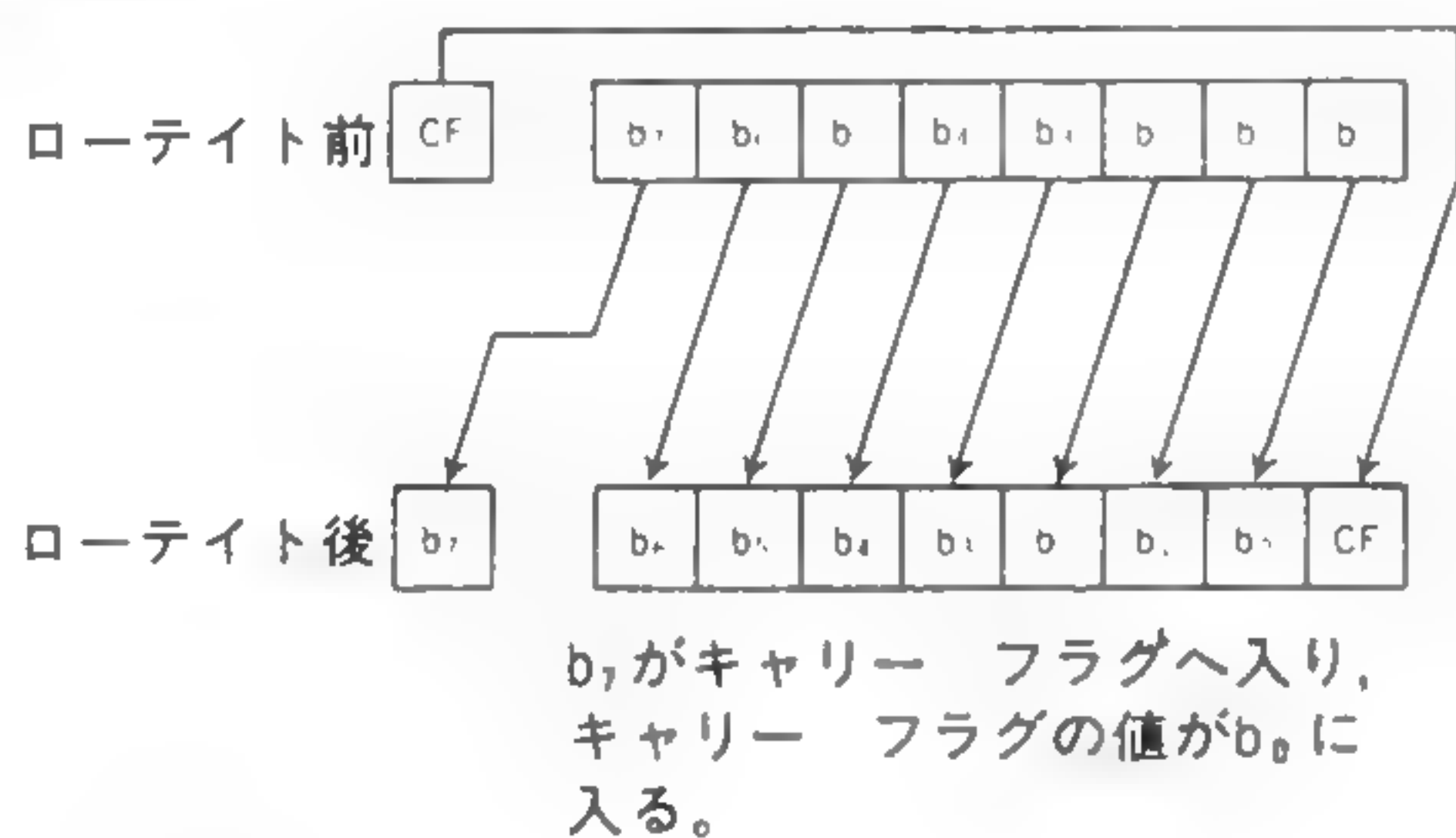
ローテイトとは、右/左へビットを回転させるもので、キャリー フラグを含めて回転させるものをたんにローテイト (Rotate)、キャリー フ

ラグを含めずに回転させるものをローテイト・サーキュラ (Rotate Circular) と呼びます。

図2-23にローテイトの、図2-24にローテイト・サーキュラの動作を示します。

図 2-23 ローテイトの動作

●左ローテイト



●右ローテイト

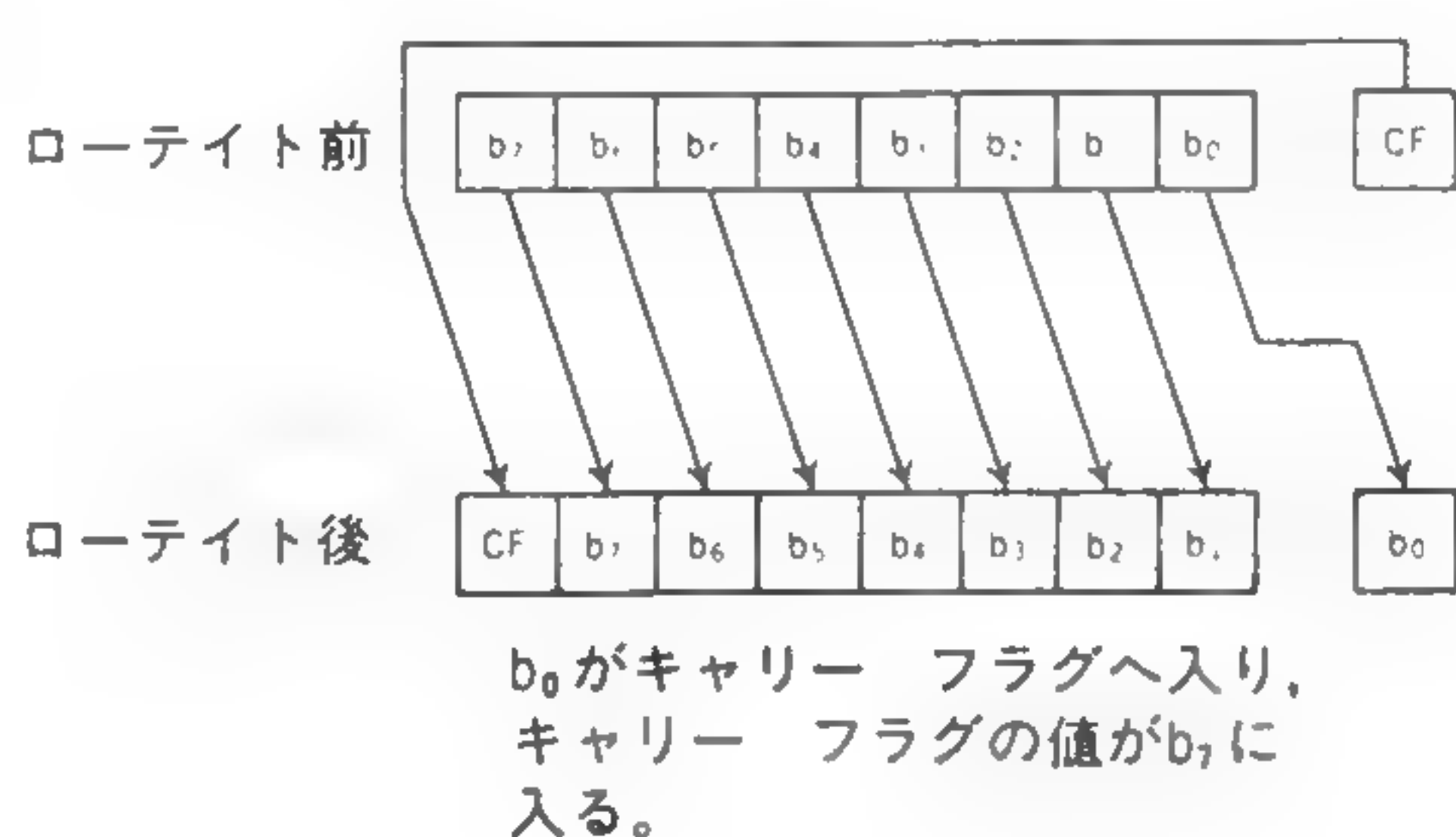
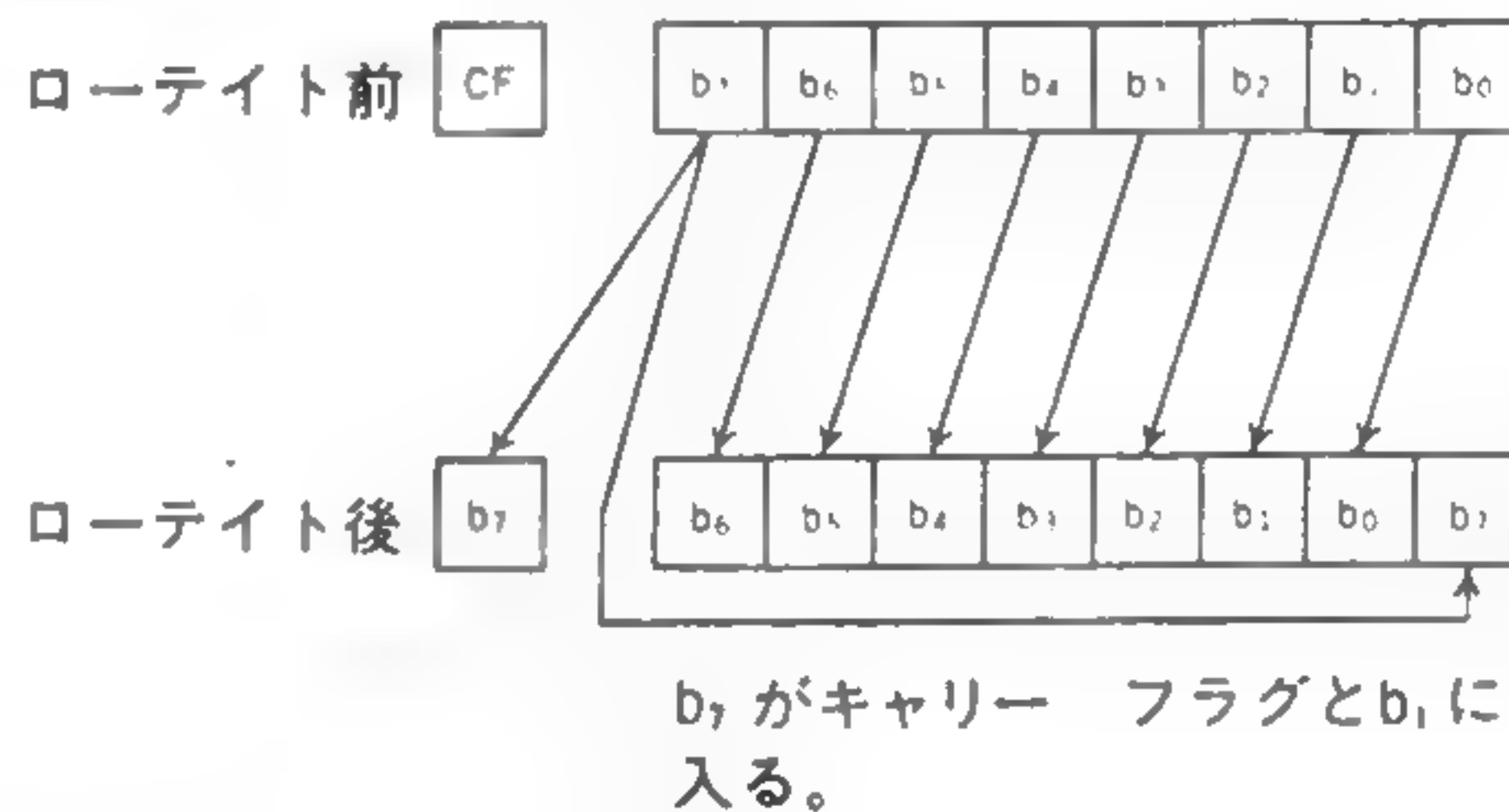
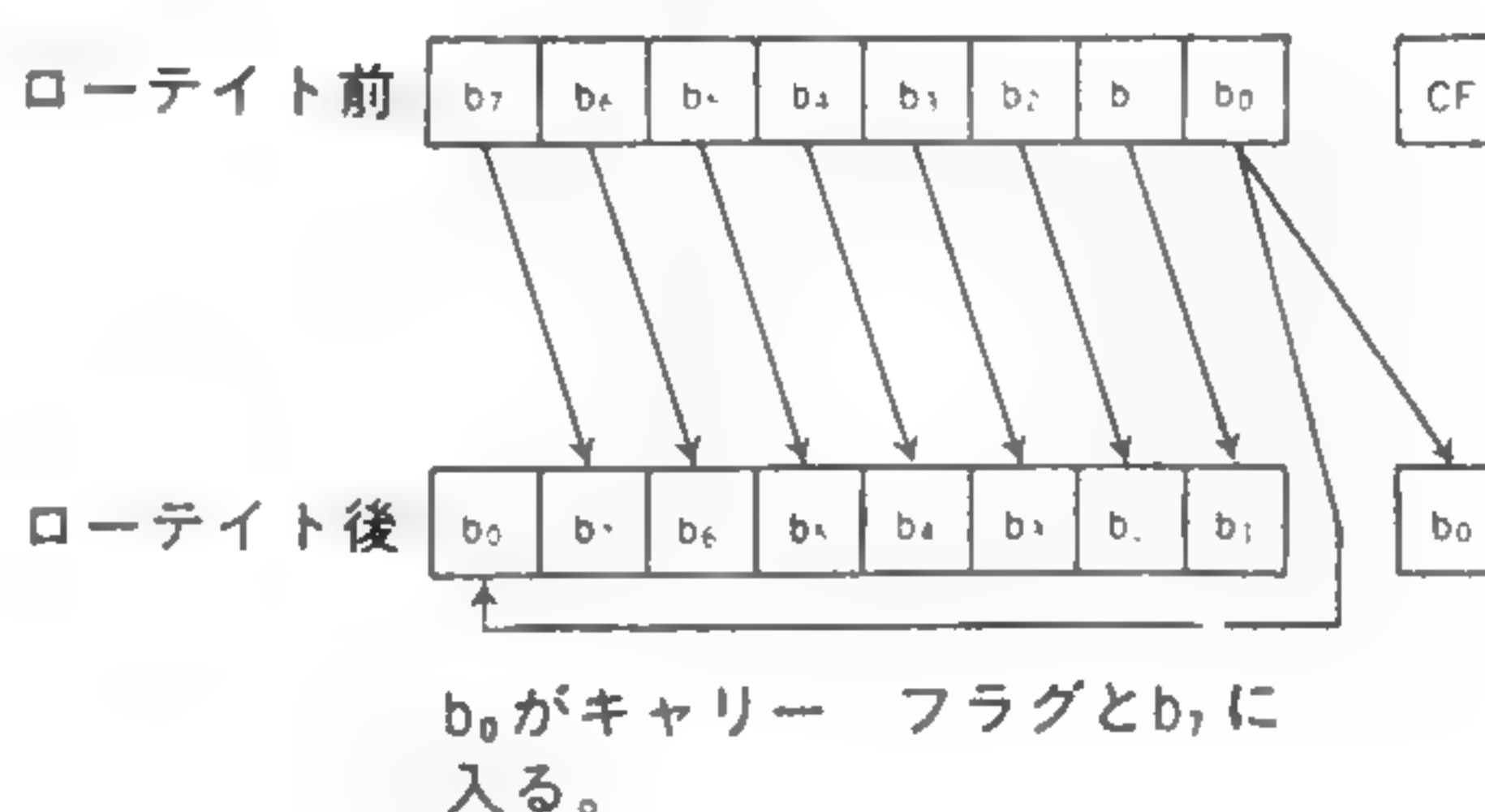


図 2-24 ローテイト・サーキュラの動作

●左ローテイト・サーキュラ



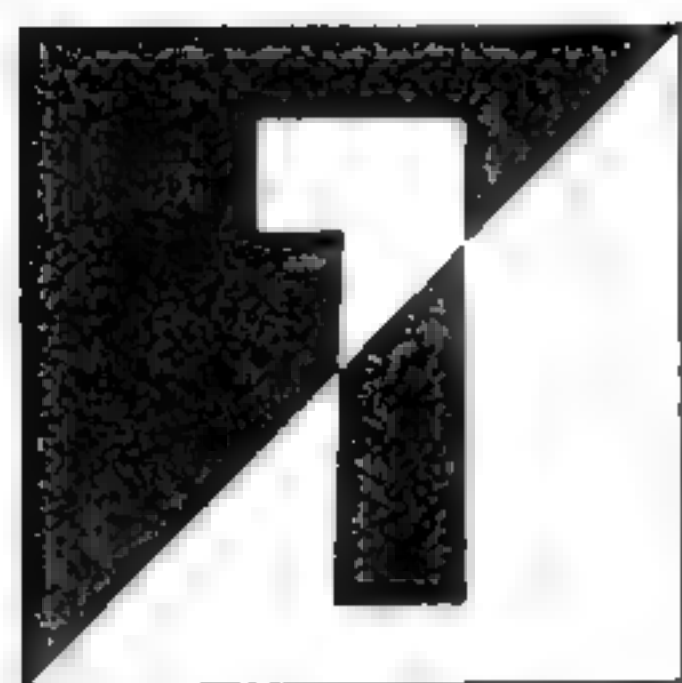
●右ローテイト・サーキュラ



A large, stylized number '5' is the central graphic. It is composed of several overlapping geometric shapes: a large circle at the base, a horizontal bar across the middle, and a pointed top. The number is rendered in a light gray color with a thin black outline. It is set against a background of diagonal gray stripes that run from the top-left to the bottom-right.

3章 Z80Aのマシン語命令

この章では、MSXのZ80ACPUのレジスタ構成や割込みなどハードウェア的なことから始め、ソフトウェアをつくる上で必要なZ80Aのニモニックとマシン語について解説していきます。



3 Z80Aのマシン語命令

Z80A CPUについて

Z80Aは米国ザイログ社が開発したマイクロ・プロセッサ(マイクロコンピュータのCPUのこと)で、米国インテル社のマイクロ・プロセッサ8080Aをベースに設計されました。Z80AファミリーにはCPUの実行速度の違いによってZ80、Z80Bなどがあります。⁽⁵⁴⁾

MSXでは、CPUのクロックに3.579545MHz⁽⁵⁵⁾を使っています。これにより、1サイクルは約279n秒(1n秒は1/10⁹秒)となります。

マシン語は、オペコード(正式にはオペレーション・コード)と呼ばれる1~2バイトの命令部(インストラクション)と、値や番地を表わすためのデータ部(1~2バイト)とで構成されています(データ部はないこともある)。したがって、Z80Aのマシン語命令は、最小1バイトから最大4バイトまであることになります。

Z80Aは、オペコードを読むのにもともと4サイクルかかります。MSXでは、さらにWA

IT(待ち時間)を1サイクル加えているため、5サイクルとなります。つまりオペコードを1命令読むのに(1サイクルの時間が約279n秒なので)5×279n秒≒1.4μ秒(1μ=1/10⁶)かかるわけです。

メモリのリード/ライトには3サイクル(3×279n秒=837n秒)、I/Oのリード/ライトにはメモリより

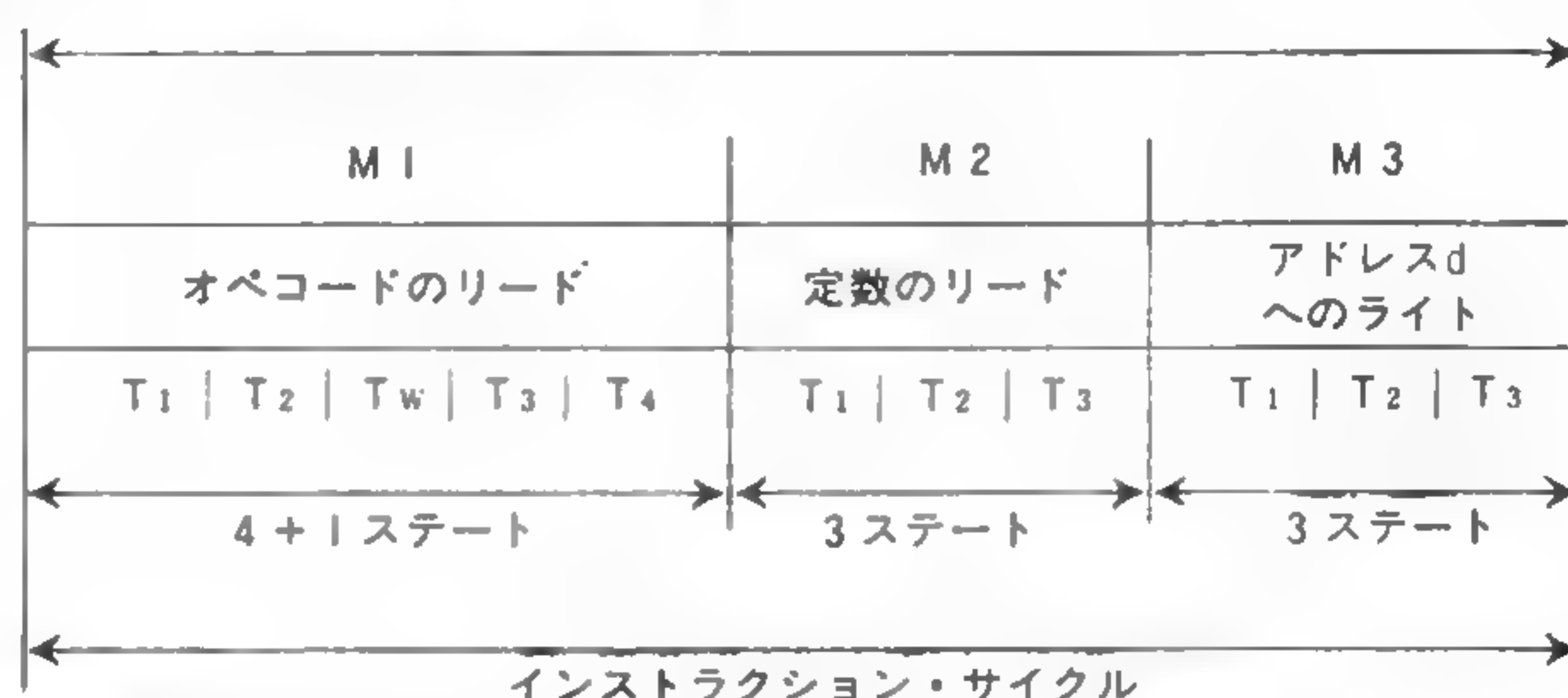
も1サイクル多い4サイクル(4×279n秒=1.116μ秒)かかります。

ここでZ80Aのマシン語命令が実際CPU内でどう処理されるのかを考えてみましょう。

例として、「オペランドがあるアドレス+1番地に記憶されている定数を、アドレスdに設定する」という命令を考えてみましょう。CPUは図3-1のように動作します。この命令の実行は、オペコードの読み取り、定数の読み取り、アドレスdへの書き込みの3つの部分に分けて行なわれます。これには、先頭からM1、M2、M3と名前がついています。これをマシン・サイクルと呼びます。この命令の場合、実行に3マシン・サイクルかかっています。

各マシン・サイクルはいくつかのステートからなり、1ステートは1サイクルで構成されています。この命令の場合、実行するのにM1が4+1ステート(+1はWAIT)、M2とM3

図3-1 CPUの命令の実行例



注) MSXマシンは、M1のとき1つWAIT(T_w)を入れています。ステート数を計算するときには注意してください。

⑤④ マイクロ・プロセッサは、水晶発振による規則的な信号を基準として動作している。この信号のことをクロックという。クロックの周波数が高いほど(信号の間隔が短いほど)、同じ命令を実行しても短時間に処理されるという長所を持っているが、それに伴いメモリやI/OのICに高速度=高価なIC

を使わなければならないという短所がある。ちなみにクロックの最高周波数は、Z80で2.5MHz、Z80Aで4MHz、Z80Bで6MHzとなっている。

⑤⑤ 注54参照。

⑤⑥ 3.579545MHzとは、1秒に3,579,545回発振するということ。

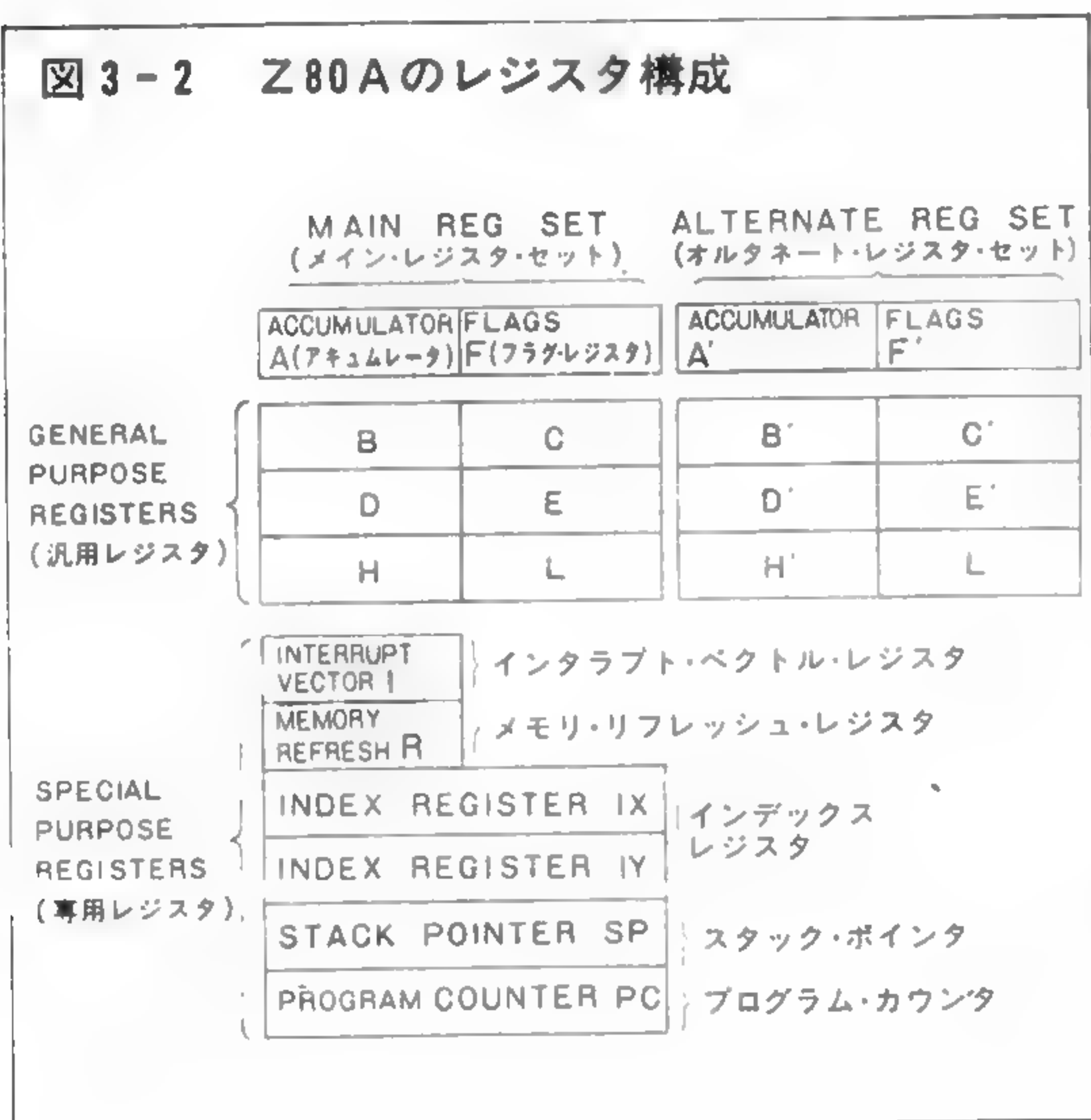
がともに3ステートなので計11ステートかかります。1つの命令の実行時間を求めるときには、このステート数によって計算します。この例では命令を1回実行するのに $11 \times 279 \text{ n秒} \div 3 \mu\text{秒}$ かかることがわかります。

マシン・サイクル、ステート数は各命令によって異なります。

1. Z80Aのレジスタ構成

Z80Aには図3-2のように18個の8ビット・レジスタと4個の16ビット・レジスタがあります。このうち汎用レジスタとして6個の8ビット・レジスタが2組(B, C, D, E, H, L, B', C', D', E', H', L')あり、この8ビットのレジスタは2つ組み合わせて16ビット・レジスタ(BC, DE, HL, BC', DE', HL')としても使うことができます。また、アキュムレータとフラグも2組(A, F, A', F')あります。

図3-2 Z80Aのレジスタ構成



1回の波の長さは $1/3579545 = 279 \text{ n秒}$ で、これが1サイクルである。

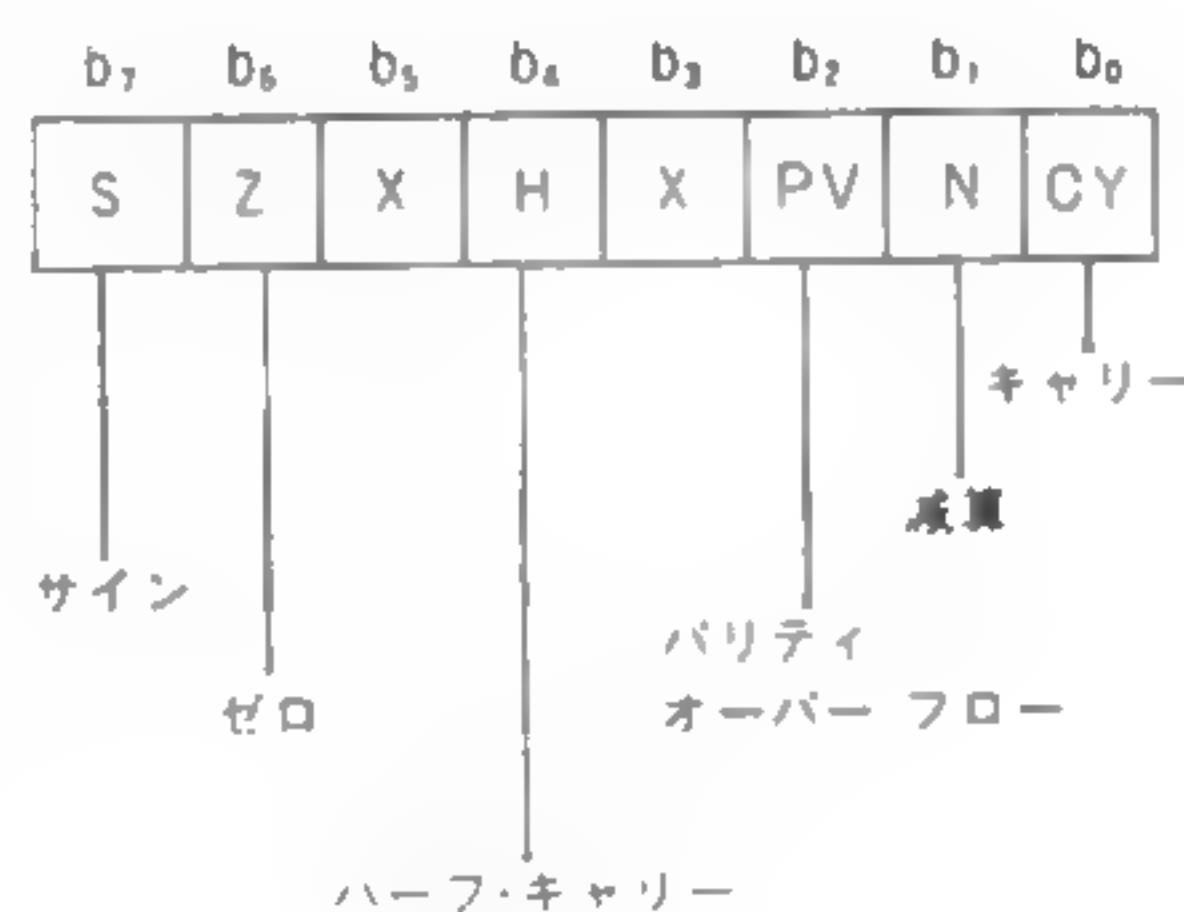
①元々「A」と「A'」という名がついているわけではなく、あくまでも使っている方をA、使っていない方をA'と表わすだけである。したがって、アセンブル・リストを読むときは、

1 アキュムレータ(A, A')とフラグ・レジスタ(F, F')

Z80Aには8ビットのアキュムレータとフラグ・レジスタのペアが2組(A, FとA', F')あります。しかし、この2組は同時に使うことができません。現在使っているアキュムレータとフラグ・レジスタをA, Fと表わし、使っていない方のアキュムレータとフラグ・レジスタをA', F'と表わします。これは1命令(EX AF, AF')⁽⁵⁷⁾で交換することができます。

8ビットの算術演算や論理演算はアキュムレータを中心に行ないます。フラグ・レジスタは、演算結果の状態を記憶するためのレジスタです。⁽⁵⁸⁾このレジスタの構成は次のようになっています。

図3-3 フラグ・レジスタ



注1)
Xは未定義フラグ
注2)
キャリー・フラグは一般にCと表わすことが多いのですが、ここでは、レジスタCと区別するためにCYとします

●キャリー フラグ(CY)

加算時にはキャリーが、減算時にはボローが入ります。また、ローテイトやシフト命令でも使われます。

●減算フラグ(N)

加算命令を実行するとリセット(0)され、減算命令を実行するとセット(1)されます。BCD演算のときに使います。⁽⁵⁹⁾

どちらのAレジスタを使用中であるのかによく注意しなければならない。

⑤⑥引算の結果がゼロであるとか、桁あふれが起きたとか。

⑤⑨BCD演算のときに、直前の演算が加算であったか減算であったかを示すのがこのNフラグである。

●パリティ/オーバーフローフラグ(P/V)

論理演算を実行すると結果のパリティを示します。パリティが偶数ならセット、奇数ならリセットされます。パリティとは、演算結果の中の1になっているビットの数のことで、偶数個(Even)か奇数個(Odd)かで表わします。

算術演算を実行すると2の補数演算のオーバーフロー(桁あふれ)を示します。演算結果の値が2の補数の範囲(-128~+127)を超えた場合セットされます。

●ハーフ・キャリーフラグ(H)

BCD演算のときに使うフラグです。⁽⁶⁰⁾

●ゼロ・フラグ(Z)

演算結果が0ならセットされ、0でなければリセットされます。

●サイン・フラグ(S)

演算結果の符号ビットが入ります。

②汎用レジスタ(B,C,D,E,H,L,B',C',D',E',H',L')

Z80Aには汎用レジスタも2組あります。この2組の汎用レジスタは、アキュムレータと同じように、同時に使うことはできません。現在使っている汎用レジスタをB,C,D,E,H,Lと表わし、使っていない方の汎用レジスタをB',C',D',E',H',L'と表わします。これも1命令(EXX)で簡単に交換できます。

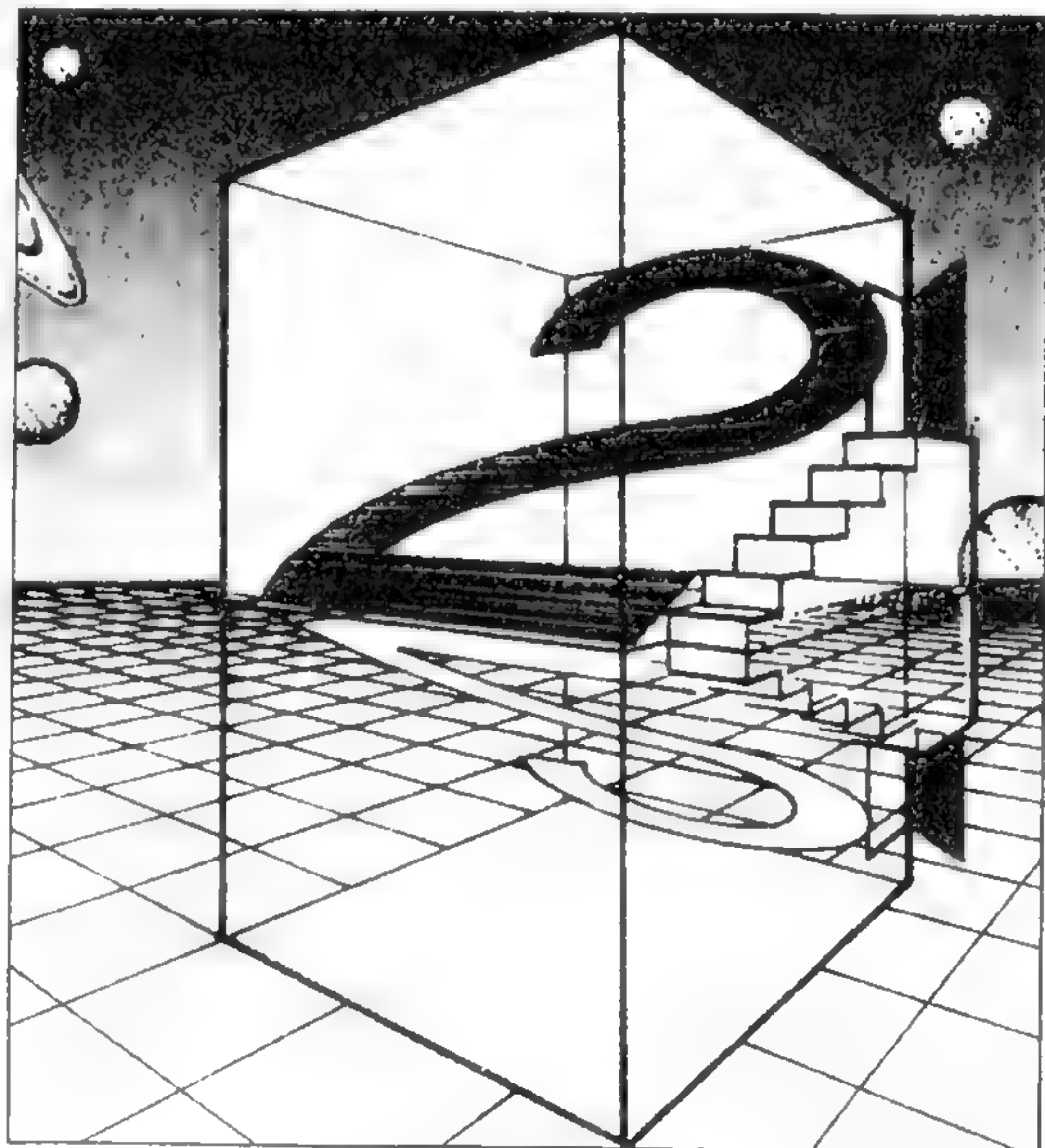
汎用レジスタは、8ビットで使うことも16ビットで使うこともできます。ただし、16ビット

で使うときはBC,DE,HLのペア・レジスタと呼ばれる単位でないといけません。

ペア・レジスタは主としてメモリやアドレスの指定(ポインタ)として用いることが多いのですが、それ以外にペア・レジスタ間で16ビットの演算(加減算だけ)ができます。その場合、ペア・レジスタのHLが16ビットのアキュムレータの役割を果たします。

③プログラム・カウンタ(PC)

プログラム・カウンタは、次に実行すべきマシン語のメモリ・アドレスを記憶しているレジスタです。PCの内容は、PCが示しているメモリ上の命令(または命令に付随した値/アドレス)を1バイト読み取るごとに自動的にインクリメント(+1)されます。⁽⁶¹⁾



⑥ BCD演算は4ビットずつを単位として計算するが、上位・下位4ビット間の桁あふれ、桁借りの有無を検出するのがHフラグである。注59のNフラグとともに、BCD演算のときに使う「DAA命令」をCPUが実行するときに内部で参照するだけで、プログラム中での条件判断には使われない。

⑦ ジャンプ命令を実行すると、インクリメントの代わりにジャンプ先のアドレスが設定される。ジャンプとは飛び越しという意味で、これを使うことでプログラムの流れを変えたり、

一部分を繰り返したりすることができる。

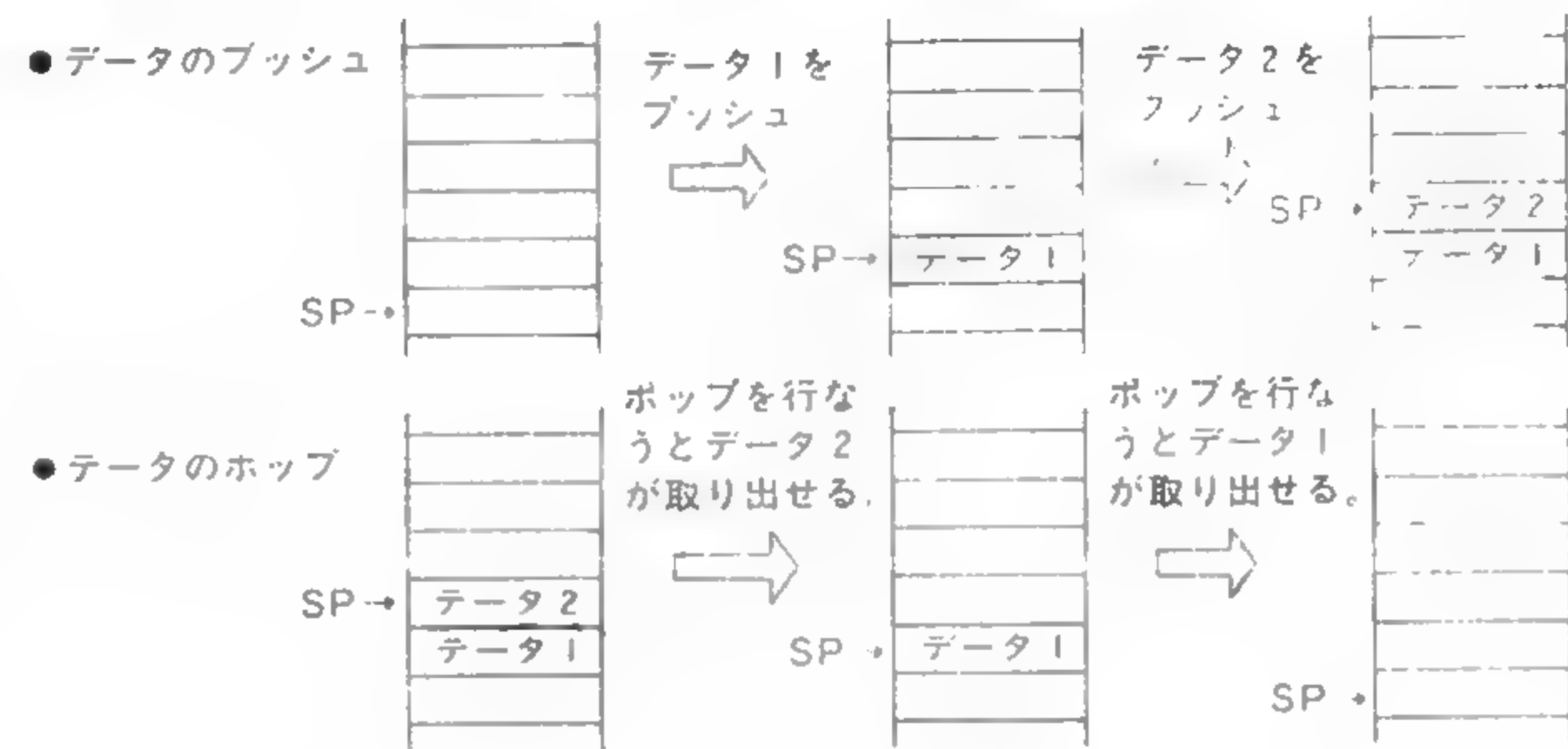
⑧ プッシュ・ダウン・スタック: プッシュ(Push)という動作でデータをスタックの上に積み重ね(実際はSPをデクリメントし、そのSPが示すアドレスにデータを入れる)、ポップ(Pop)という動作でスタックの一番上のデータを取る(実際はSPが示すアドレスのデータを読み、SPをインクリメントする)というように、最後に入れたデータを最初に取り出せるようなスタックのことをいう。

4 スタック・ポインタ(SP)

スタック・ポインタは、メモリ内にあるプッシュ・ダウン・スタック⁽⁶²⁾の先頭アドレスを記憶するレジスタです。(図3-4参照)

スタックとは「積み重ねる」という意味で、この場合データやアドレスを一時的に記憶するためのメモリ領域のことです。

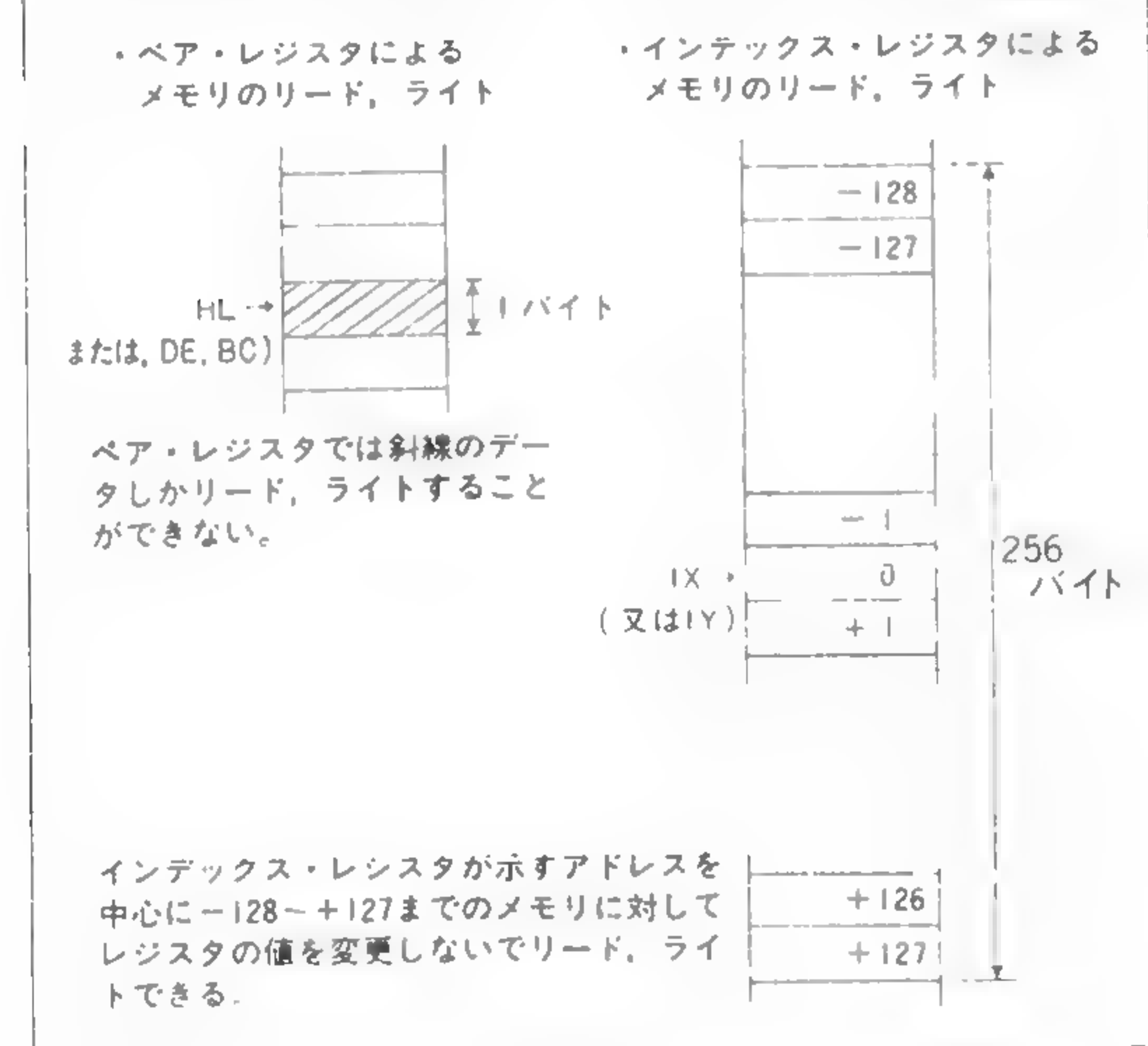
図3-4 プッシュ・ダウン・スタックの動作



5 インデックス・レジスタ(IX, IY)

Z80AはIXとIYという2本のインデックス・レジスタを持っています。このレジスタは、ペア・レジスタと同じくメモリ・アドレスの指定(ポインタ)として使います。インデックス・レジスタをポインタとして使えば、自身の内容を変更せずに-128バイト~+127バイト⁽⁶³⁾までのアドレスを指定することができます(図3-5)。

図3-5 インデックス・レジスタとペア・レジスタの違い



6 インタラプト・ベクトル・レジスタ(I)

インタラプト・ベクトル・レジスタは、MSX⁽⁶⁴⁾では使いません。Z80Aには割込みのモードが3種類あり、Iレジスタはそのうちのモード2と呼ばれる割込みで使うのですが、MSXではモード1という割込みモードしか使いませんので、このレジスタは何の用途にも使われていません。しかし、プログラムでデータの記憶などにこのレジスタを使うことは避けた方がよいでしょう。

7 メモリ・リフレッシュ・レジスタ(R)

メモリ・リフレッシュ・レジスタは、図3-1(34ページ)で示したM1のT3、T4ステートのとき、ダイナミックRAM⁽⁶⁵⁾というメモリのためにリフレッシュ・アドレス⁽⁶⁶⁾というものを出力します。このレジスタだけは7ビット構成で、M1ごとに自動的にインクリメントされます。このレジスタについては、このようなものがあるということだけ覚えておけばよいでしょう。

⑥③ 同じアドレスを指定するのにも、(HL+1)と指定できないのでHLレジスタの内容自体を+1しなければならずこれではHLの内容が変わってしまうが、(IX+1)とすればIXレジスタ自体の内容は変わらずにアドレスを指定できる

⑥④ 38ページ参照

⑥⑤ ダイナミックRAM: RAMにはダイナミックRAM(DRAM)とスタティックRAM(SRAM)の2種類がある。ダイナミックRAMは、一定時間ごとに記憶内容を復習させ

てやらないと忘れてしまうのに対し、スタティックRAMは電源を供給している限り覚えている。ダイナミックRAMに復習させてやることをリフレッシュといい、普通はDRAMをリフレッシュするための特別なハードウェアが必要であるが、Z80Aからはこのための信号が出ており大変簡単にDRAMを扱うことができる。リフレッシュする間隔は品番や容量によっても違うが、約2ms以下でなければならない。

⑥⑥ 次にどのメモリ・ブロックをリフレッシュするかを示す。

2. 割込みについて

Z80Aにはノンマスクابل(NMI)とマスクابل(INT)の2種類の割込みがあります。⁶⁷⁾そのうちマスクابلの割込みには前にも述べたように3種類の割込みモードがあります。⁶⁸⁾このモードはハードウェアによって決まるもので勝手に変更することはできません。

MSXではノンマスクابلの割込みは使われていません。マスクابلの割込みではモード1を使っています。モード1の割込みでは、割込みが発生したとき、戻り番地(リターン・アドレス)をスタックにプッシュし、0038(16進)番地にジャンプします。

割込み(インタラプト:Interrupt)は、実行中のプログラムをハードウェア的に中断し特定のルーチン(あるまとまった機能を実行するための一連の命令群)を実行します。この特定のルーチンを割込みルーチンとかインタラプト・ルーチンと呼びます。これを図にしたのが図3-6です。割込みルーチンでは初めにCPU内部のレジスタを退避させ、終わりにそのレジスタを復帰させています。割込みルーチン内ではこの操作を必ず行なわなければなりません。そうしないと割込みが発生する前のプログラムに戻ったとき、レジスタの内容が元通りでないためにプログラムが正しく実行されなくなってしまう。

また、決まった番地(0038H)からしか始めることができないモード1の割込みでは、複数の入出力装置からの割込み要求を以下のようにして判定・実行しています。

ハードウェア側では、すべての割込み要求線をOR(論理和)で接続しています。こうすると、接

続している装置(または回路)のどれか1つでも割込みを要求すればすぐ割込みが発生します。

しかしこの段階では、まだどの装置(回路)からの割込み要求だったかはわかりません。割込みルーチン側では、図3-7のようにI/Oポートを調べていき、割込みを要求している装置(回路)を見つけにいきます。見つかったら、その装置(回路)に対して割込み処理を行ないます。

MSXでは、同じような方法によってインターバル・タイマ(一定時間ごとに割込みを要求する回路、VDPにこの機能がある)や、外部スロットからの割込み要求を処理しています。

図3-6 割込みが発生したときの流れ

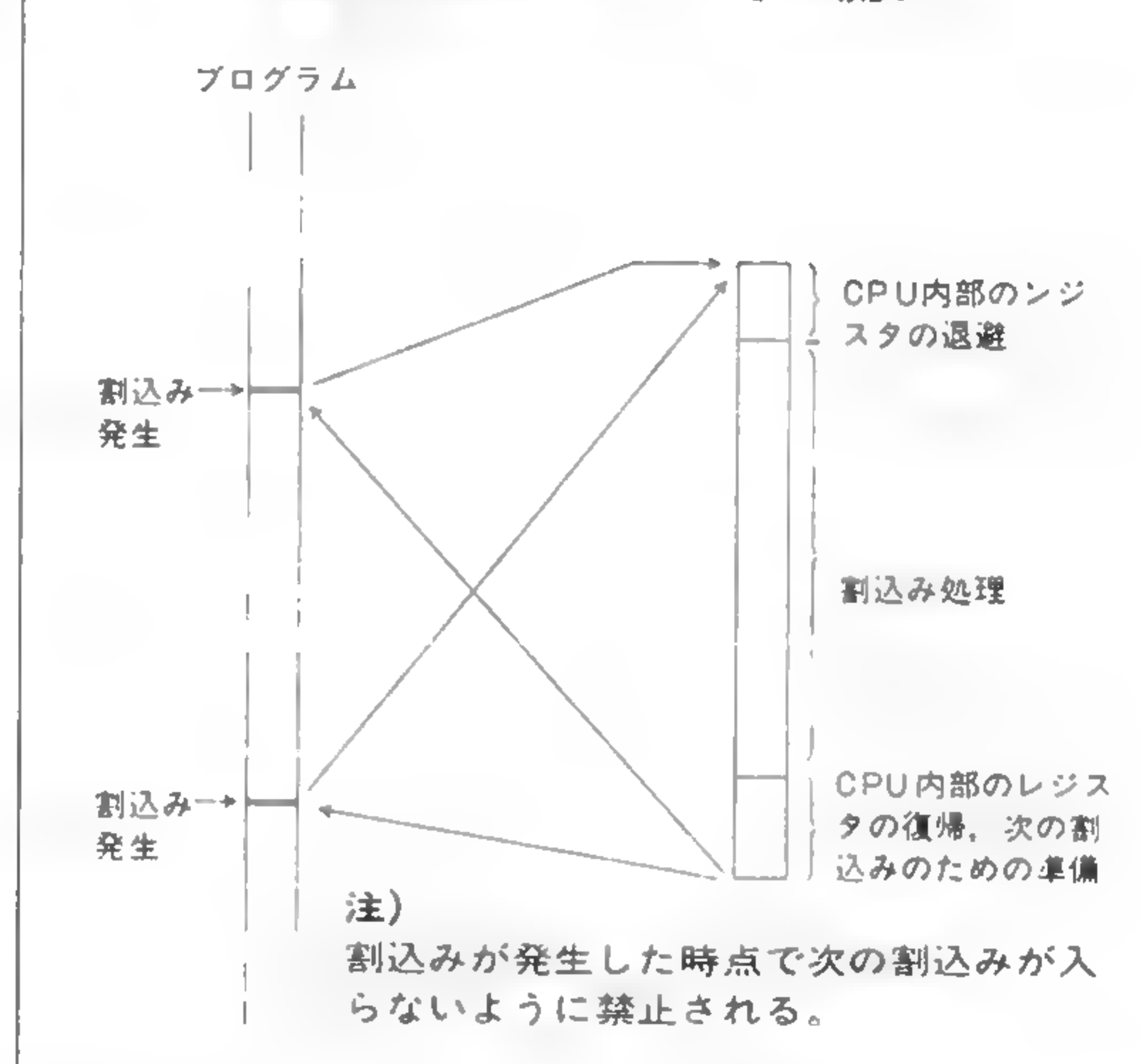
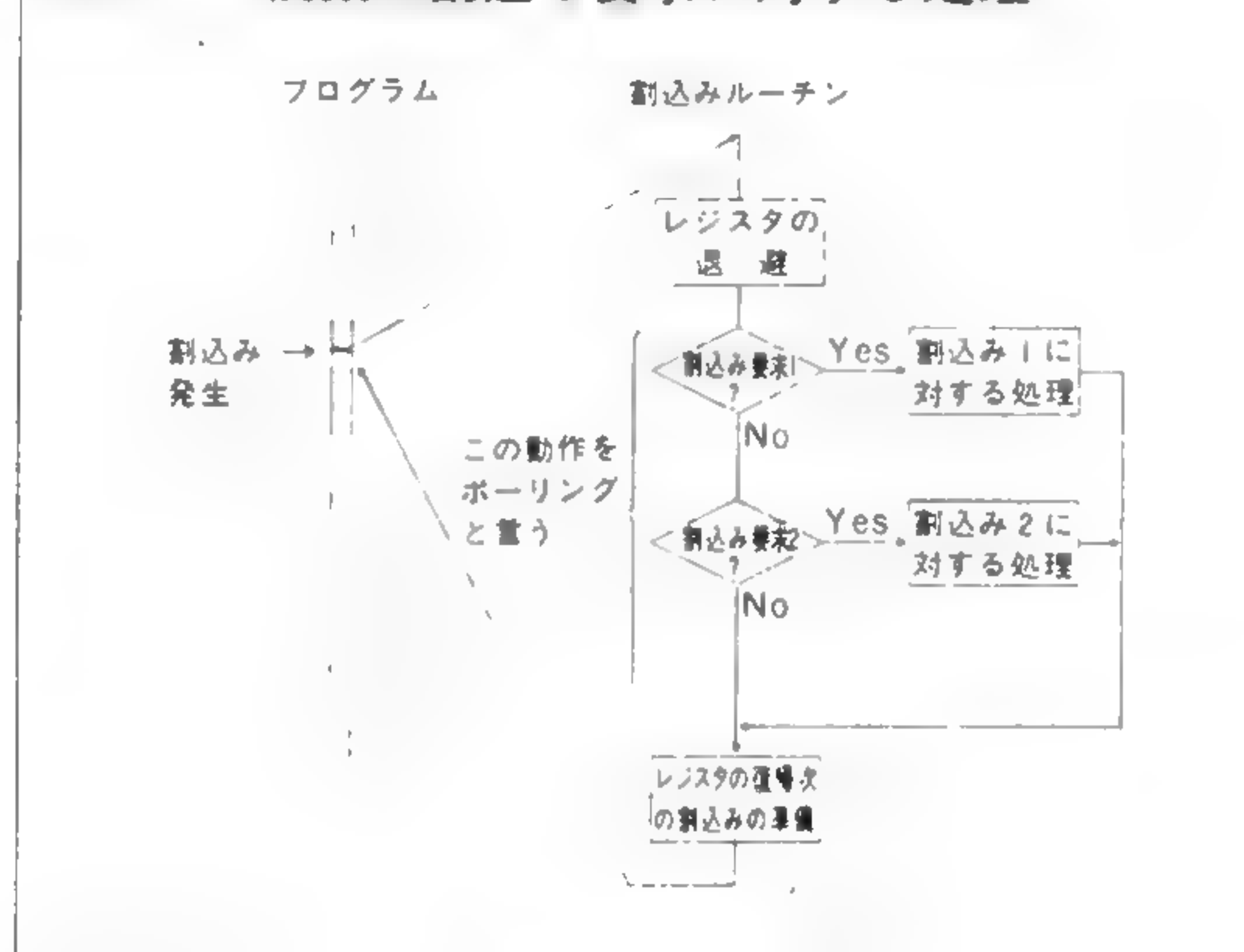


図3-7 複数の割込み要求に対する処理



⁶⁷⁾ Z80Aでは、割込みはハードウェアによって起こる。ノンマスクابلはプログラムで禁止できない。⁶⁸⁾ 37ページ参照。



Z80Aのニモニックとマシン語

Z80Aには、マシン語命令が158種類、696命令もあります。命令自体は1バイトから4バイトの数値であるため、この組み合わせで人間がプログラムをつくるのは大変なことです。そこでプログラムをつくるときは数値でなく、人間にわかりやすい単語を組み合わせることにします。⁶⁹たとえば「アキュムレータに16進数の5Fを入れる」という命令ならば「LD A, 5FH」と書くわけですが（このような書きかたをアセンブラ言語という）。これは「3E, 5F」という2バイトのマシン語（マシン・コードともいう）に対応します。

ここで「LD A, 5FH」のLDとはLoad（ロードとはコンピュータの分野では『～へ入れる』といった意味に使われる）の略で、AはアキュムレータAを示します。5FHは数値で、5FのあとにHをつけることによって16進数であることを示します。また、このような数値（アドレスも含む）は必ず0～9の数字で始めるようにします。そうしないと、たとえば16進数のAなどはアキュムレータAと区別できないからです。⁷⁰

アセンブラ言語は、ニモニックとオペランドに分けられます。ニモニックとは「LD」のように動作を表わすもの、オペランドとは「A, 5FH」のようにレジスタやメモリ、アドレス、I/Oアドレス、数値などの細かい指定をするものです。ニモニックによってはオペランドを持たないものもあります。

Z80Aのニモニックは次の67種類があります。

①データの転送、交換

LD, PUSH, POP, EX, EXX

②ブロック転送とブロック・サーチ

LDIR, LDDR, LDI, LDD

CPIR, CPDR, CPI, CPD

③演算

ADD, SUB, ADC, SBC, CP, INC, DEC

NEG, AND, OR, XOR, CPL, DAA

④ローテイト、シフト

RLC, RRC, RL, RR, SLA, SRA, SRL

RLD, RRD

RLCA, RRCA, RLA, RRA

⑤ビット操作、フラグ操作

BIT, RES, SET, SCF, CCF

⑦ジャンプ、コール、リターン

JP, JR, DJNZ, CALL, RST

RET, RETI, RETN

⑦入出力

IN, INI, INIR, IND, INDR

OUT, OUTI, OTIR, OUTD, OTDR

⑧CPUコントロール

NOP, HALT, DI, EI, IM

アドレッシングやニモニックを解説する前にオペランドについて次の規則を決めておきます。

- ①数値、アドレスは0～9の数字で始まり、16進数は最後にHをつける。Hがつかないものは10進数とみなす。

例) 0F32H...16進数のF32, 1234...10進数の1234

- ②オペランドが2つあるときは、初めのオペランドを第1オペランド、次のオペランドを第2オペランドと呼ぶ。

⁶⁹ 5 ページ参照。 ⁷⁰ 16進数のAを意味したいときは、0Aと入れる。

1. Z80Aのアドレッシング・モード

アドレッシング・モードとは、メモリなどをリード/ライトするときのアドレスやレジスタの指定方法のことです。メモリ以外に、I/OやCPU内のレジスタも対象にしています。

Z80Aには10種類のアドレッシング・モードがあります。の中には、ある特定のニモニックでしか使われないものから、複数のニモニックで共通に使われるものまであります。

Z80Aのアドレッシングは次の10種類です。

①レジスタ・アドレッシング

(Register Addressing)

これはCPU内のレジスタをアクセス先として指定するものです。指定できるレジスタは、A,B,C,D,E,H,Lの7つの8ビット・レジスタと、AF,BC,DE,HL,SP,IX,IYの7つの16ビット・レジスタです。

8ビット・レジスタは英字1文字、16ビット・レジスタは英字2文字で表わします。たとえばオペランドに「B」と書かれていればレジスタB、「HL」と書かれていればペア・レジスタHLを示します。



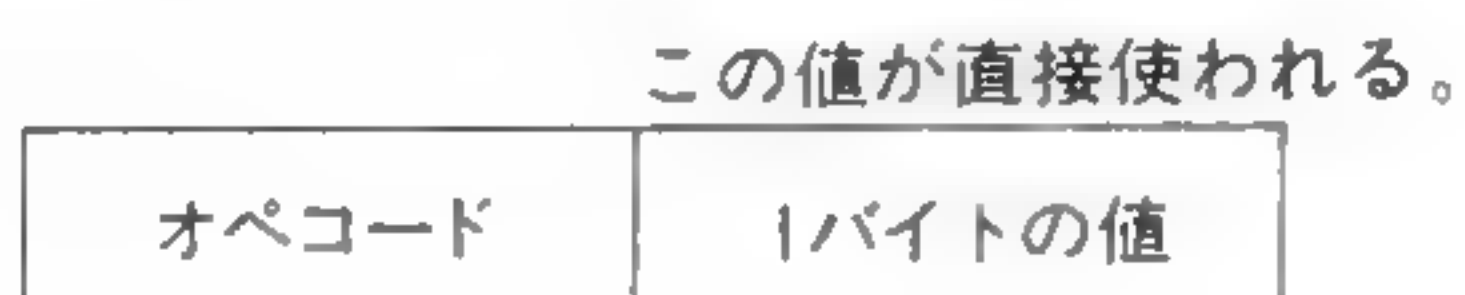
②イミディエイト・アドレッシング

(Immediate Addressing)

マシン語命令で直接指定された1バイトの値をソース（送り側）とするものです。

オペランドには1バイトで表わせる数値(10進数で0～255または－128から＋127、16進数で0Hから0FFH)を記述します。

マシン語では次のようになります。

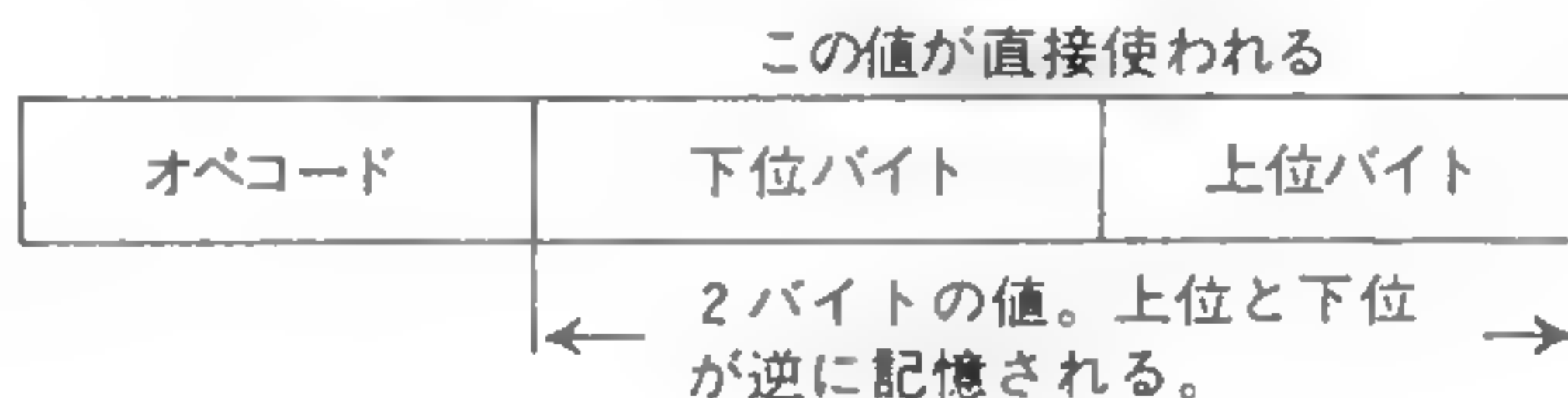


③イミディエイト・エクステンド・アドレッシング (Immediate Extended Addressing)

マシン語命令で直接指定された2バイトの値をソース（送り側）とするものです。

オペコードには2バイトで表わされる数値(10進数で0～65535または－32768～＋32767、16進数で0H～0FFFFH)を記述します。

マシン語では次のようになります。

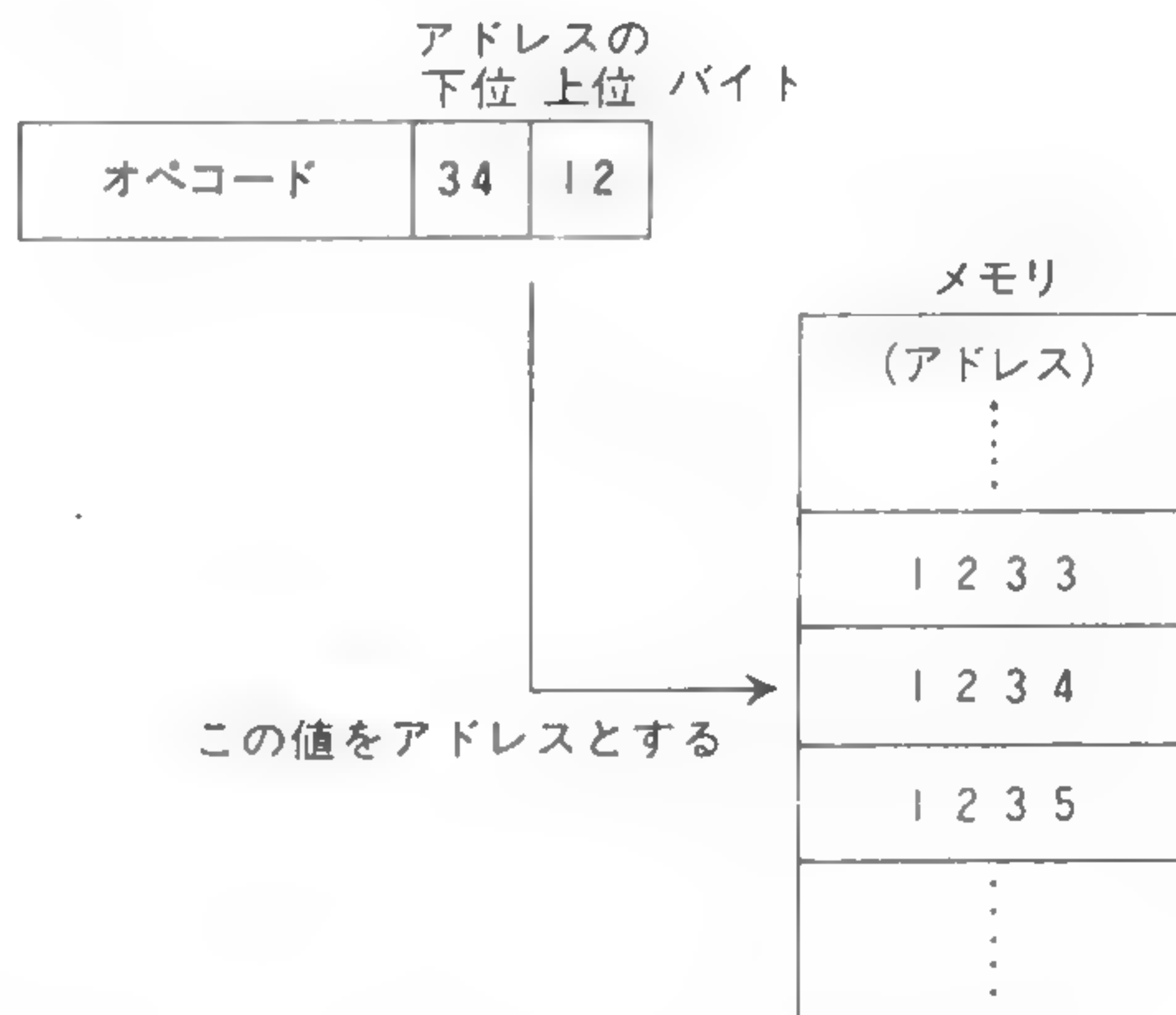


④エクステンド・アドレッシング

(Extended Addressing)

これは、絶対アドレッシングのことで、マシン語命令で直接指定された2バイトの値をメモリのアドレス(0Hから0FFFFH)とします。

オペランドには、アドレス(2バイトの数値)をカッコ()で囲んだものを記述します。たとえば(1234H)と書かれていれば、次のようになります。



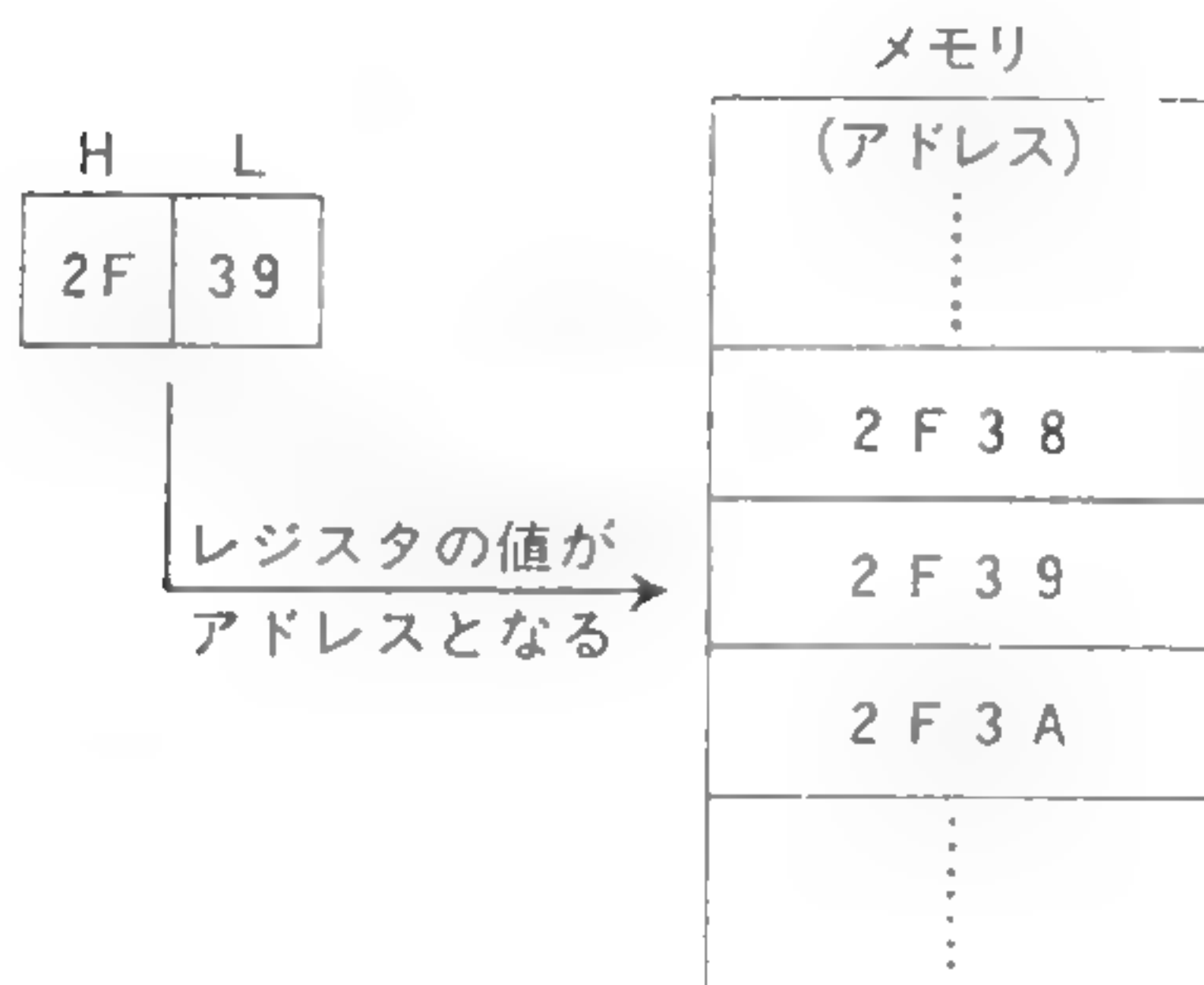
⑤レジスタ・インダイレクト・アドレッシング

(Register Indirect Addressing)

16ビット・レジスタ（ペア・レジスタまたはIX、IYなど）でメモリのアドレスを指定する方法です。16ビット・レジスタの名をカッ

コで囲んだものを記述します。

たとえば、(HL)と書かれていれば、次のようになります。



⑥ インデックス・アドレッシング

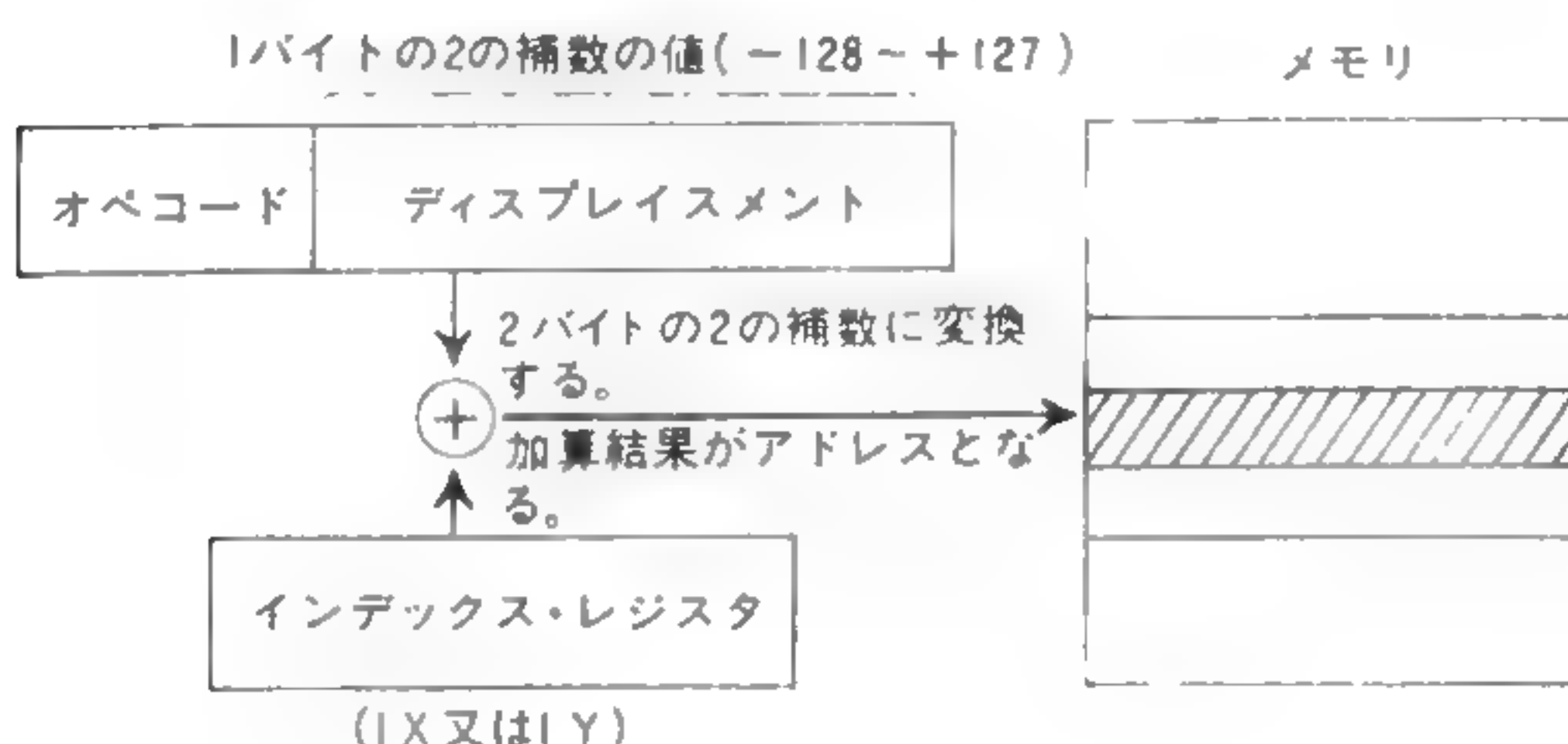
(Indexed Addressing)

インデックス・レジスタ (IX, IY) を使ったインデックス修飾でメモリのアドレスを指定するモードです。インデックス・アドレッシングでは、マシン語命令で直接指定された1バイトの数値をディスプレイメントといいます。インデックス修飾とはインデックス・レジスタの値とディスプレイメントの値を2の補数で加算し、求められた値をメモリのアドレスとする方法です。

インデックス・アドレッシングは次のように記述します。

(IX + ディスプレイメント) または
(IY + ディスプレイメント)

ディスプレイメントには-128~+127までの数値を、たとえば (IX + 26), (IY - 128) のように書きます。実アドレスは次のように求めます。



⑦ レラティブ・アドレッシング

(Relative Addressing)

命令で直接指定した1バイトのディスプレイメントを、プログラム・カウンタ (PC) に2の補数で加算し、求められた値をメモリのアドレスとする方法です。このアドレッシングはジャンプ命令以外では使えません。

⑧ ビット・アドレッシング

(Bit Addressing)

レジスタやメモリのビットを直接指定するアドレッシングです。このアドレッシングはビット操作命令以外では使えません。

⑨ モディファイ・ページ・ゼロ・アドレッシング

(Modified Page Zero Addressing)

このアドレッシングは、リスタート命令 (RST) だけに使われるものです。

⑩ インプライ・アドレッシング

(Implied Addressing)

CPUの特定レジスタを自動的に選択する情報を命令中に含むアドレッシングです。

以上、10種類のうち①~⑥のアドレッシングは複数のニモニックで使われ、オペランドで指定されます。⑦~⑩のアドレッシングは、特定のニモニックでしか使われません。詳しくは各ニモニックのところで説明します。

2. データの転送, 交換

データの転送に関する命令には、

LD (LDとは「Load」の略)

PUSH

POP

があります。ロード(LD)命令には8ビット・ロード命令と16ビット・ロード命令があります。PUSH命令とPOP命令は、16ビット・レジス

タの内容をスタックにプッシュ、ポップする命令です。

データ交換に関する命令には、
EX (EXとは「E x c h a n g e」の略)
EXX

があります。**E X**命令は16ビット・レジスタに対する命令で、**E X X**命令は汎用レジスタに対する命令です。

1 ロード命令

ニモニックは「LD」で、2つのオペランドを

持っています。第1オペランドがデスティネーション (Destination：受け側)、第2オペランドがソース (Source：送り側) です。転送はソースからデスティネーションに向かって行なわれます。^{①)}

ロード命令は、
LD デスティネーション, ソース
と表わします。

① 8ビット・ロード命令

表3-1は8ビット・ロード命令のマシン・コードの一覧表です。この表は、使われるアド

表3-1 8ビット・ロード命令 “LD des, sou”

SOURCE DESTINATION		IMPLIED		REGISTER							REG INDIRECT			INDEXED		EXT. ADDR.	IMME.
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n
REGISTER	A	ED 5 7	ED 5 F	7 F	7 8	7 9	7 A	7 B	7 C	7 D	7 E	0 A	1 A	DD 7 E d	FD 7 E d	3 A n n	3 E n
	B			4 7	4 0	4 1	4 2	4 3	4 4	4 5	4 6			DD 4 6 d	FD 4 6 d		0 6 n
	C			4 F	4 8	4 9	4 A	4 B	4 C	4 D	4 E			DD 4 E d	FD 4 E d		0 E n
	D			5 7	5 0	5 1	5 2	5 3	5 4	5 5	5 6			DD 5 6 d	FD 5 6 d		1 6 n
	E			5 F	5 8	5 9	5 A	5 B	5 C	5 D	5 E			DD 5 E d	FD 5 E d		1 E n
	H			6 7	6 0	6 1	6 2	6 3	6 4	6 5	6 6			DD 6 6 d	FD 6 6 d		2 6 n
	L			6 F	6 8	6 9	6 A	6 B	6 C	6 D	6 E			DD 6 E d	FD 6 E d		2 E n
REG INDIRECT	(HL)			7 7	7 0	7 1	7 2	7 3	7 4	7 5							3 6 n
	(BC)			0 2													
	(DE)			1 2													
INDEXED	(IX+d)			DD 7 7 d	DD 7 0 d	DD 7 1 d	DD 7 2 d	DD 7 3 d	DD 7 4 d	DD 7 5 d							DD 3 6 d n
	(IY+d)			FD 7 7 d	FD 7 0 d	FD 7 1 d	FD 7 2 d	FD 7 3 d	FD 7 4 d	FD 7 5 d							FD 3 6 d n
EXT. ADDR.	(nn)			3 2 n n													
IMPLIED	I			ED 4 7													
	R			ED 4 F													

注) ・転送は SOURCE から DESTINATION へ行なわれる
 ・dはディスプレイスメント (2の補数-128 ~+127)

・nは1バイトの値 ・nnは2バイトの値、メモリ上には上位バイトと下位バイトが逆に記憶される。

①すなわちオペランドの右側から左側へロードが行なわれる。
 ②表3-3は「LD A, R」,「LD A, I」を実行したときのフラグの変化を示す。この命令を実行すると P Vフラグが変化し、割込みが許可されているのが禁止されているのかわかる。

レッシング別に分けられています。表において第1オペランドのデスティネーション側は縦線の左側に示し、第2オペランドのソース側は横線の上部に示してあります。この表で空欄になっているアドレッシングの組み合わせは、CPUが実行できない命令を示します。たとえば「LD A, B」という命令なら「7 8 H」とマシン語命令が求められますが、「LD D, (BC)」という命令は表において空欄になっているのでこの命令はない（実行できない）ことを表わします。

表3-2は8ビット・ロード命令のマシン・サ

イクル数とステート数の一覧表です。ステート数はZ80Aの標準値とカッコ内にMSXでのステート数を示します。MSXは、前にも述べたようにM1サイクルのときWAITを1つ入れるので標準値より1ステート多くなっています。オペコードが2バイトの命令はM1が2回あるのと同じなので、ステート数が標準値より2つ多くなっています。

8ビット・ロードは「LD A, R」, 「LD A, I」の2つの命令を除きフラグは変化しません。

表3-2 8ビット・ロード命令のマシン・サイクル数とステート数

SOURCE DESTINATION		IMPLIED		REGISTER							REG INDIRECT			INDEXED		EXT. ADDR.	IMME.			
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n			
REGISTER	A	M = 2 T = 9 (11)		マシン・サイクル M = 1 ステート T = 4 (5)							M = 2 T = 7 (8)			M = 5 T = 19 (21)		M = 4 T = 13 (14)		M = 2 T = 7 (8)		
	B																			
	C																			
	D																			
	E																			
	H																			
	L																			
REG INDIRECT	(HL)			M = 2 T = 7 (8)												M = 3 T = 10 (11)				
	(BC)																			
	(DE)																			
INDEXED	(IX+d)			M = 5 T = 19 (21)												M = 5 T = 19 (21)				
	(IY+d)																			
EXT. ADDR.	(nn)			M=4 T=13 (14)																
IMPLIED	I			M=2 T=9 (11)																
	R																			

注) ・ Mはマシン・サイクル
・ Tはステートのこと。
・ カッコ内の数はMSXの場合のステート数。

表3-3 ロード命令「LD A, R」, 「LD A, I」によるフラグの変化

命 令	フラグ・レジスタ						
	S	Z	H	P/V	N	CY	
LD A, R; LD A, I	↑	↑	×	0	×	↑	IFF (0: 割込み禁止 1: 割込み許可)

注)
・ =実行結果の影響を受けない。
0 =リセットされる。 × =不定
↑ =実行結果の影響を受ける。

②16ビット・ロード命令

ニモニックは8ビット・ロード命令と同じ「LD」ですが、8ビットか16ビットかはオペランドで区別されます。

表3-4は16ビット・ロード命令の一覧表です。
表3-5は16ビット・ロード命令のマシン・サイクル数とステート数です。

表3-4 16ビット・ロード命令“LD des, sou”

SOURCE DESTINATION		REGISTER							IMM. EXT.	EXT. ADDR.
		AF	BC	DE	HL	SP	IX	IY	nn	(nn)
REGISTER	AF									
	BC								0 1 n n	E D 4 B n n
	DE								1 1 n n	E D 5 B n n
	HL								2 1 n n	2 A n n
	SP					F 9	DD F 9	FD F 9	3 1 n n	E D 7 B n n
	IX								DD 2 1 n n	DD 2 A n n
	IY								FD 2 1 n n	FD 2 A n n
EXT.ADDR.	(nn)		E D 4 3 n n	E D 5 3 n n	2 2 n n	E D 7 3 n n	DD 2 2 n n	FD 2 2 n n		

注) ・転送はSOURCEからDESTINATIONへ行なわれる。
・nnは2バイトの値
メモリ上には上位バイトと下位バイトが逆に記憶される。

表3-5 16ビット・ロード命令のサイクル数とステート数

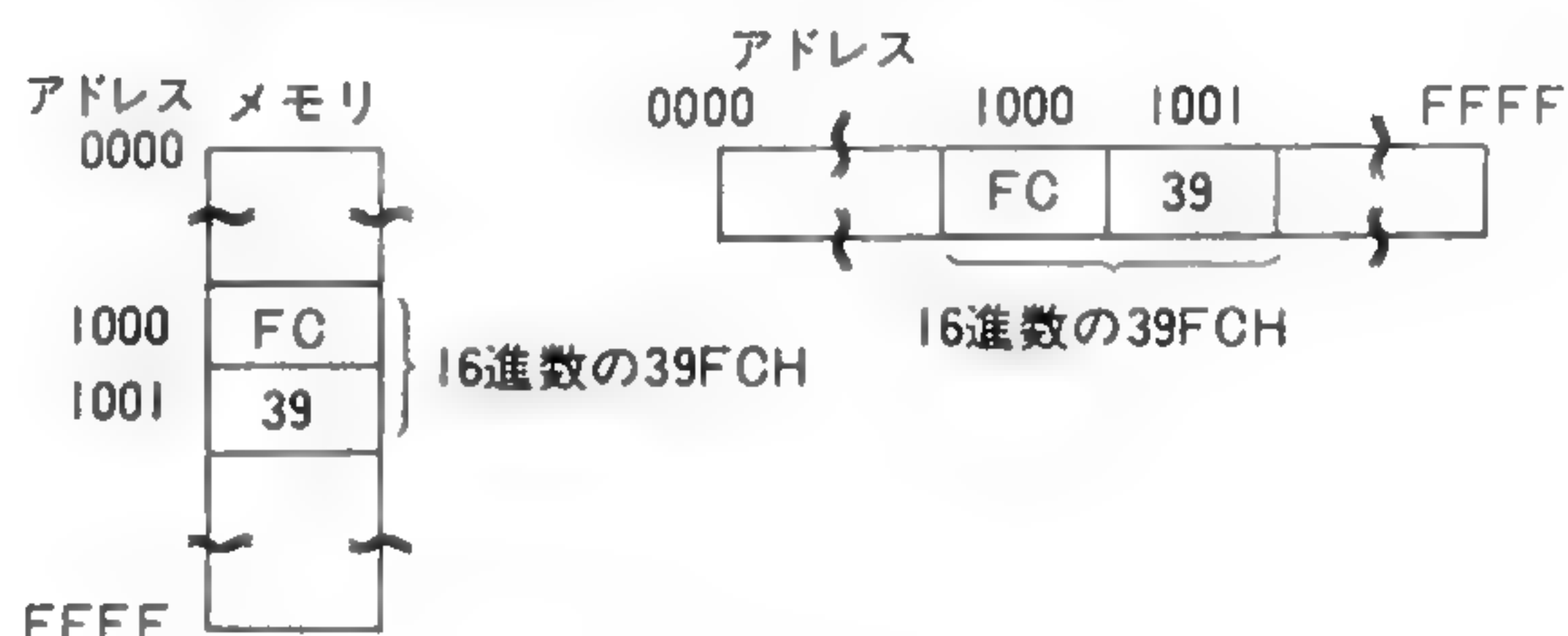
SOURCE DESTINATION		REGISTER							IMM. EXT.	EXT. ADDR.
		AF	BC	DE	HL	SP	IX	IY	nn	(nn)
REGISTER	AF									
	BC									M=6 T=20 (22)
	DE									M=3 T=10 (11)
	HL									M=5 T=16 (17)
	SP					M=1 T=6 (7)	M=2 T=10(12)			
	IX								M=4 T=14 (16)	M=6 T=20 (22)
	IY									
EXT.ADDR.	(nn)		M=6 T=20(22)	M=5 T=16 (17)		M=6 T=20(22)				

注) ・Mはマシン・サイクル
・Tはステートのこと。
・カッコ内の数はMSXの場合のステート数。

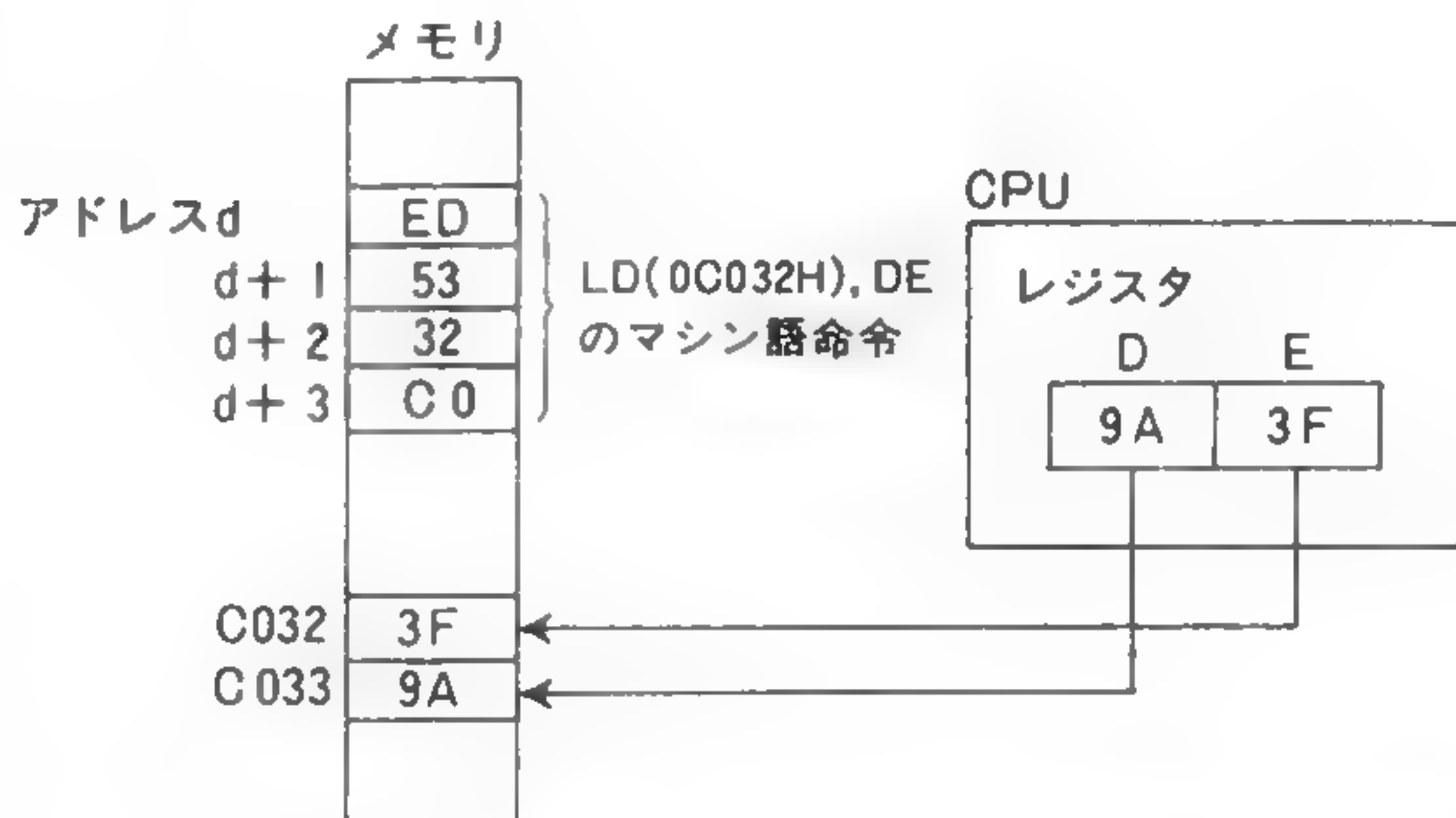
16ビット・ロード命令はフラグには影響を与えません。

Z80Aは、16ビット（2バイト）のデスティネーションをメモリ上に記憶するとき、アドレスが大きい方に上位8ビットを記憶し、小さい方に下位8ビットを連続して記憶します。この場合メモリ上には上位8ビットと下位8ビットが逆になって格納されます。

たとえば「39FCH」という16ビットのデータがメモリの1000H、1001H番地に記憶されているときは次のようになります。



これを「LD (0C032H), DE」という命令で見てみましょう。



以上のように上位、下位を逆に記憶する現象は16ビット・ロード命令以外にも、エクステンデッド・アドレッシングやイミディエイト・エクステンデッド・アドレッシングを使っている命令のアドレスやイミディエイト・データについてもいえます。

PUSH命令, POP命令

表3-6にPUSH命令, POP命令のマシン・コードとマシン・サイクル数, ステート数を示します。

PUSH命令には次の6つがあります。

PUSH AF PUSH BC PUSH DE
 PUSH HL PUSH IX PUSH IY

どれもフラグは変化しません。

PUSH命令の動作を図3-8に示します。

POP命令には次の6つがあります。

POP AF POP BC POP DE
 POP HL POP IX POP IY

フラグは「POP AF」を除き変化しません。「POP AF」では、スタックから戻したデータがアキュムレータ(A)とフラグ(F)に入るため、当然フラグが変化します。

POP命令の動作を図3-9に示します。

表3-6 PUSH, POP 命令の
 マシン・コードとマシン・サイクル、ステート数

		PUSH "PUSH reg"			POP "POP reg"		
		マシン コード	マシン サイクル	ステート	マシン コード	マシン サイクル	ステート
REGISTER	AF	F 5	3	11 (12)	F 1	3	10 (11)
	BC	C 5			C 1		
	DE	D 5			D 1		
	HL	E 5			E 1		
	IX	DD E 5	4	15 (17)	DD E 1	4	14 (16)
	IY	FD E 5			FD E 1		

注) ステートのカッコ内の数はMSXの場合のステート数

図3-8 PUSH命令の動作

例として (PUSH AF
 PUSH HL) を行なったときのスタックの様子を示します。

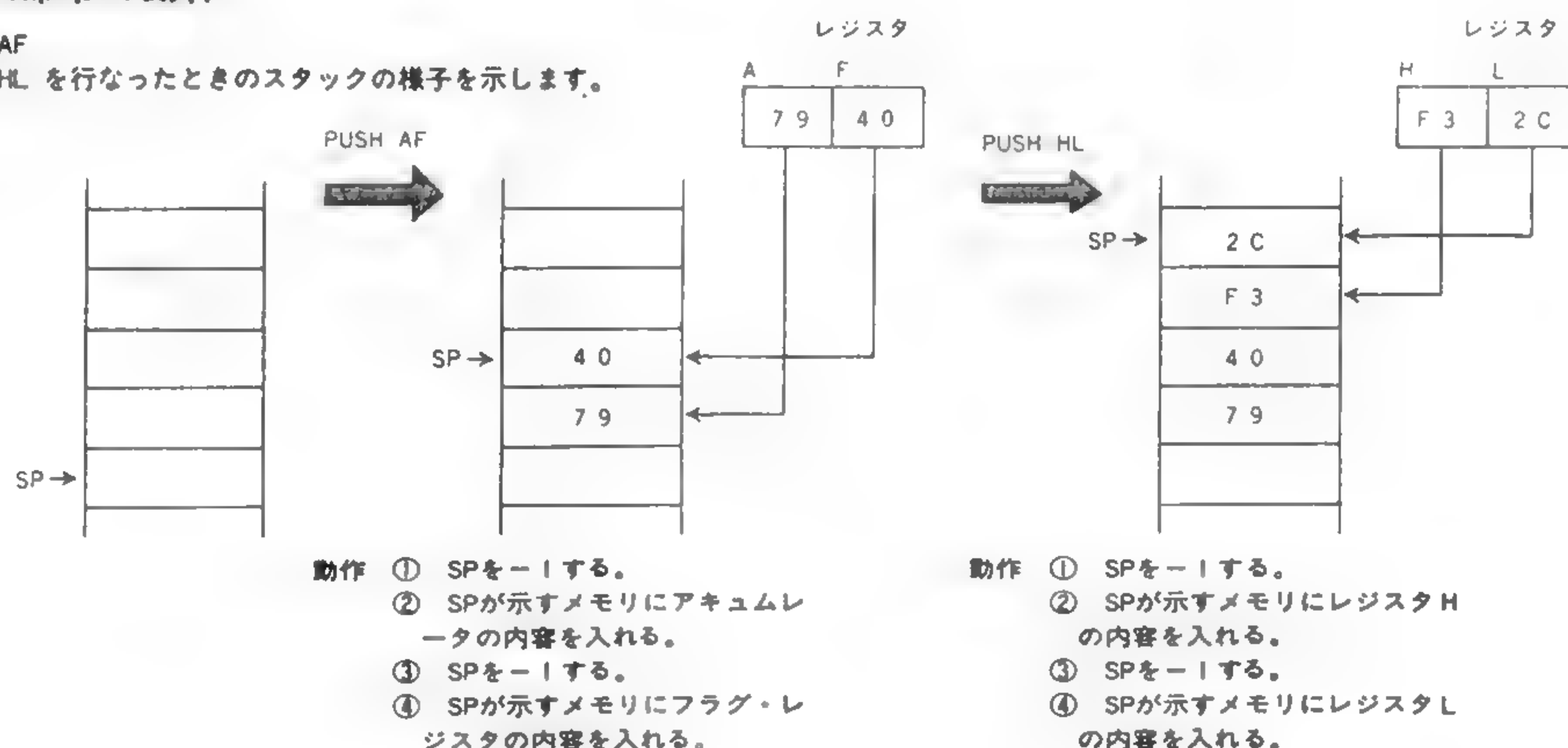
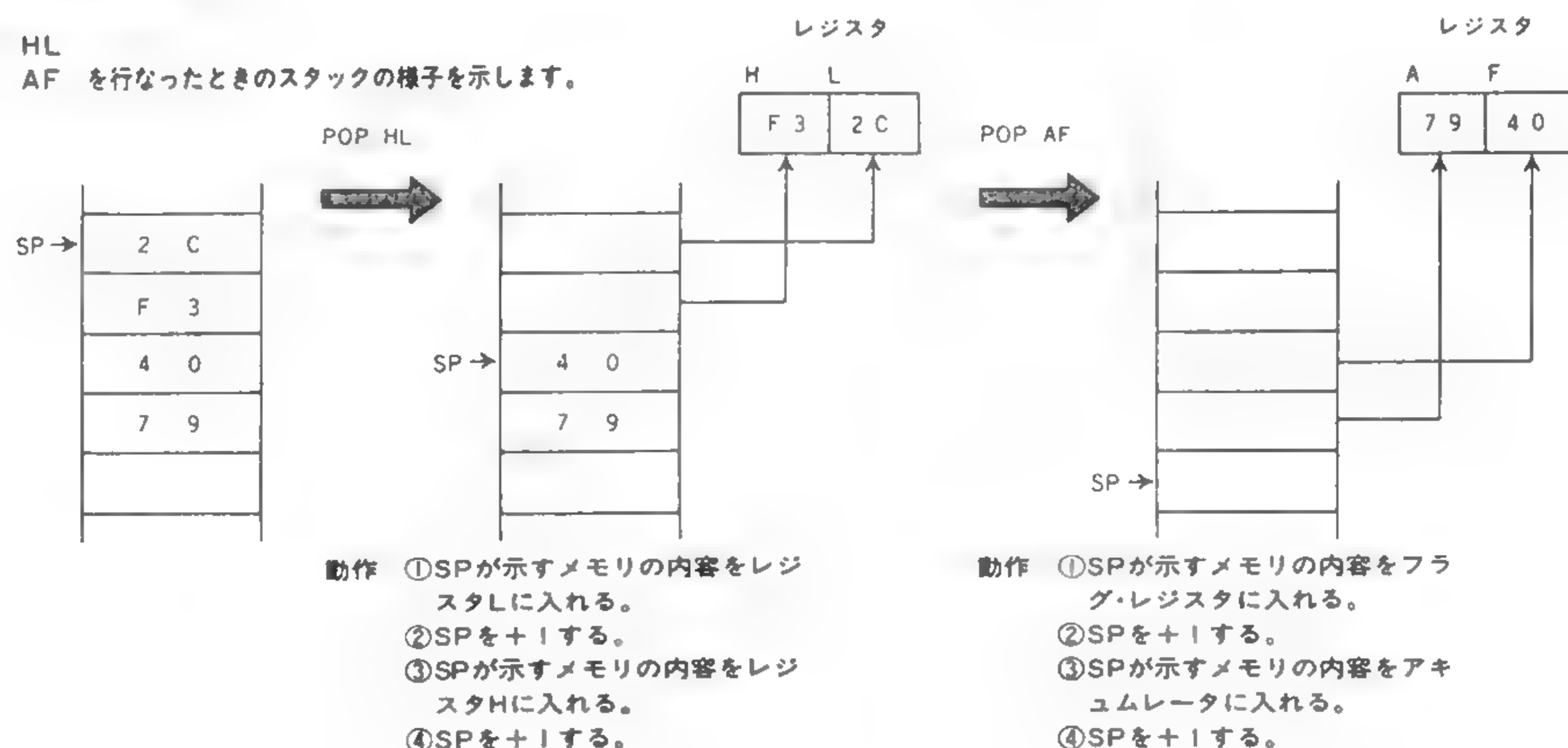


図3-9 POP命令の動作

例として (POP HL
 POP AF) を行なったときのスタックの様子を示します。



4 データの交換命令

データの交換命令には次の6つがあります。

EX AF,AF' EXX EX DE,HL
EX (SP),HL EX (SP),IX EX (SP),IY

表3-7, 8, 9にEX命令, EXX命令のマシン・コード, マシン・サイクル数, ステート数を示します。

「EX AF, AF'」命令はメイン・レジスタ側のアキュムレータ (A) とフラグ (F) をオルタネート・レジスタ側のレジスタ (A',F') と交換する命令です。

たとえばA=12H, F=40H, A'=3FH, F'=01Hのときこの命令を実行すると, A=3FH, F=01H, A'=12H, F'=40Hとなります。

「EXX」命令はオペランドがありません。こ

の命令はメイン・レジスタ側の汎用レジスタ (B, C,D,E,H,L) とオルタネート・レジスタ側の汎用レジスタ (B',C',D',E',H',L') の内容を交換する命令です。

「EX DE, HL」命令はペア・レジスタDEとHLの内容を交換する命令です。たとえば, DE=1234H, HL=9C92H のときこの命令を実行すればDE=9C92H,HL=1234Hとなります。「EX (SP), HL」,「EX (SP), IX」,「EX (SP), IY」の命令はスタック・トップの2バイトのデータとペア・レジスタHLまたはインデックス・レジスタ (IX, IY) の内容を交換する命令です (図3-10)。

データの交換命令では「EX AF, AF'」を除きフラグは変化しません。

表3-7 EX命令“EX opr1,opr2”

第1オペランド \ 第2オペランド		IMPLIED			
		AF'	HL	IX	IY
IMPLIED	AF	08			
	DE		EB		
REG.INDIR.	(SP)		E3	DD E3	FD E3

表3-8 EX命令のマシン・サイクル数とステート数

第1オペランド \ 第2オペランド		IMPLIED			
		AF'	HL	IX	IY
IMPLIED	AF	M=1 T=4 (5)			
	DE		M=1 T=4 (5)		
REG.INDIR.	(SP)		M=5 T=19 (20)	M=6 T=23(25)	

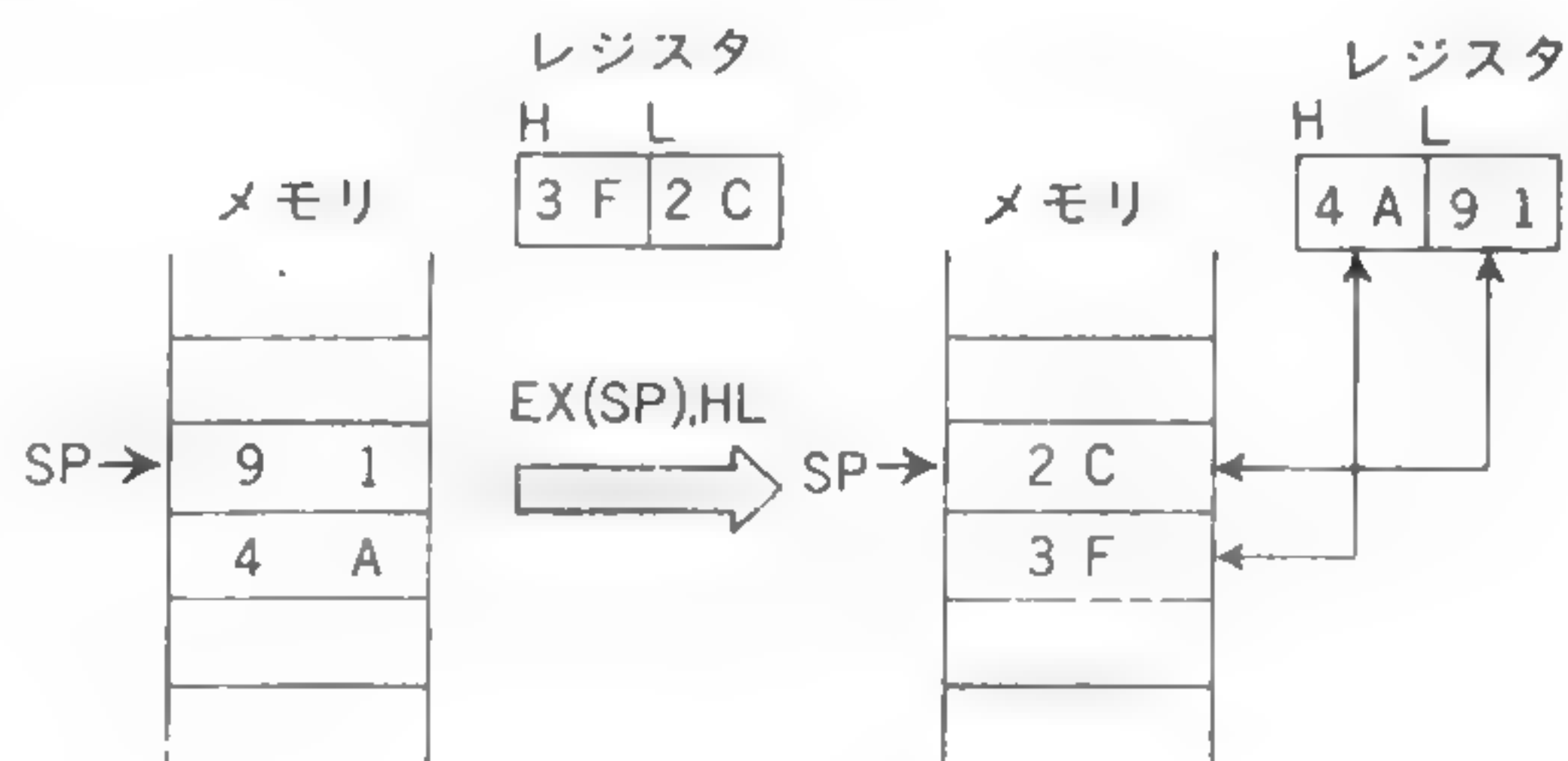
注) ・Mはマシン・サイクル
・Tはステートのこと。
・カッコ内の数はMSXの場合のステート数

表3-9 EXX命令

	“EXX”			
	マシン コード	マシン サイクル	ステート	動作
IMPLIED	D9	1	4 (5)	BC ↔ BC' DE ↔ DE' HL ↔ HL'

注) ・ステートのカッコ内の数はMSXの場合のステート数。

図3-10 「EX (SP), HL」の動作



動作 ①SPが示すメモリの内容とレジスタLの内容を交換する。
②SPが示す次のアドレスの内容とレジスタHの内容を交換する。

注) 「EX (SP), IX」, 「EX (SP), IY」も同様な動作を行なう。

● 2組のレジスタのうち、現在使っていない方のレジスタ。裏レジスタともいう。詳しくは3章1.1(35ページ)参照。

3. ブロック転送, ブロック・サーチ

ロード (LD) 命令は1命令で1バイト~2バイトのデータをレジスタ-メモリ間で転送するだけでしたが、ここで紹介する**ブロック転送命令**は、メモリ上の大量のデータを1命令で別のアドレスにそっくり転送することができる命令です。

ブロック転送に関する命令は、
 LDIR (LoaD,Increment and Repeat の略)
 LDDR (LoaD,Decrement and Repeat の略)
 LDI (LoaD and Increment の略)
 LDD (LoaD and Decrement の略)
 があります。

ブロック・サーチ命令は、メモリ上にある大量のデータの中から特定のデータを検索(サーチ)するための命令です。

ブロック・サーチに関する命令には、
 CPIR (ComPare,Increment and Repeatの略)
 CPDR (ComPare,Decrement and Repeatの略)
 CPI (ComPare and Increment の略)
 CPD (ComPare and Decrement の略)
 があります。

ブロック転送,ブロック・サーチ命令にはどちらもオペランドがありません。

1 ブロック転送命令

ブロック転送命令は次のペア・レジスタを特定の目的で使います。

- BC : バイト数カウンタ
- DE : データの転送先のアドレス
- HL : 転送するデータの格納アドレス

これらのペア・レジスタの内容は転送実行前に設定しておきます。

表3-10にマシン・コード,マシン・サイクル数,

ステート数を示し、表3-11にブロック転送命令を実行したときのフラグの変化を示します。

表3-10 ブロック転送命令(LDIR, LDDR, LDI, LDD)

	マシンコード	マシンサイクル	ステート	動作
"LDIR"	ED B 0	BC≠0のとき 5	BC≠0のとき 21 (23)	Load (DE) ← (HL) Inc HL & DE, Dec BC, Repeat until BC=0
"LDDR"	ED B 8	BC=0のとき 4	BC=0のとき 16 (18)	Load (DE) ← (HL) Dec HL & DE, Dec BC, Repeat until BC=0
"LDI"	ED A 0	4	16 (18)	Load (DE) ← (HL) Inc HL & DE, Dec BC
"LDD"	ED A 8			Load (DE) ← (HL) Dec HL & DE, Dec BC

注) ステートのカッコ内の数はMSXの場合のステート数。

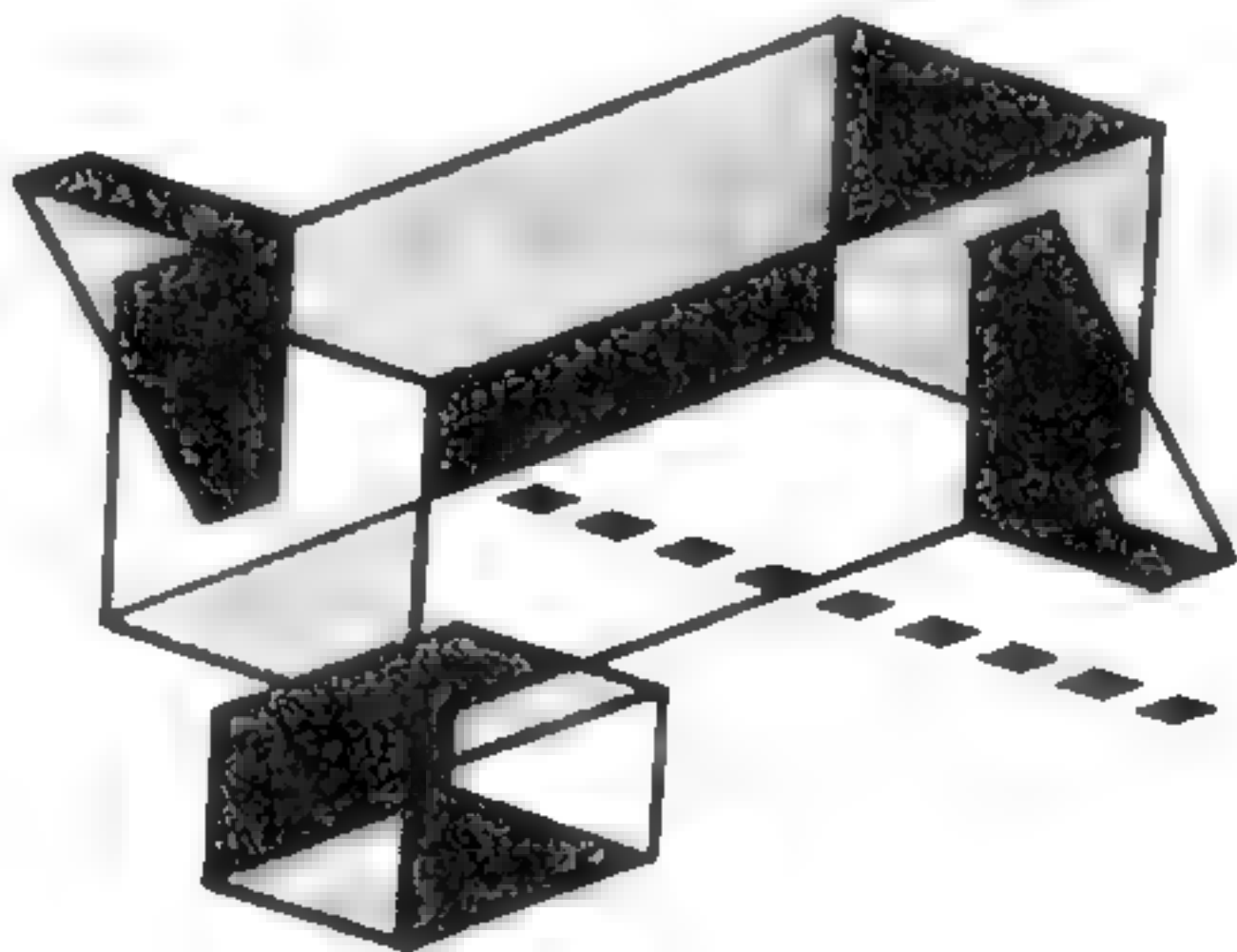
表3-11 ブロック転送命令によるフラグの変化

命 令	フラグ・レジスタ							
	S	Z	H	P/V	N	CY		
LDIR ; LDDR	•	•	X	0	X	0	0	•

LDI ; LDD	•	•	X	0	X	↑	0	•
-----------	---	---	---	---	---	---	---	---

↑
 もしBC-1=0 ならば
 P/V=0, 其他 P/V=1

注) • =実行結果の影響を受けない。
 0 =リセットされる。
 X =不定
 ↑ =実行結果の影響を受ける。



①LDIR 命令

LDIR 命令を実行すると、HLが示すメモリからデータを読み出し、DEが示すメモリに転送します。1バイトのデータ転送後HLとDEはインクリメント(+1)され、BCはデクリメント(-1)されます。BCの値がゼロになるまでこの動作を続けます。

この動作を図にすると図3-11のようになります。

②LDDR 命令

LDDR 命令を実行すると、HLが示すメモリからデータを読み出し、DEが示すメモリに転送します。データ転送後HL、DE、BCはデクリメント(-1)されます。BCの値がゼロになるまでこの動作を繰り返します。

この動作を図にすると図3-12のようになります。

③LDIR 命令とLDDR 命令の使い分け

転送する領域と転送される領域が重なっている場合、その重なりかたによってLDIR 命令、LDDR 命令を使い分けます。重なりがまったくない場合は使いやすい方の命令を使えばよいでしょう。

図3-13にLDIR 命令を使う場合、図3-14にLDDR 命令を使う場合を示します。

図3-11 LDIR命令の動作

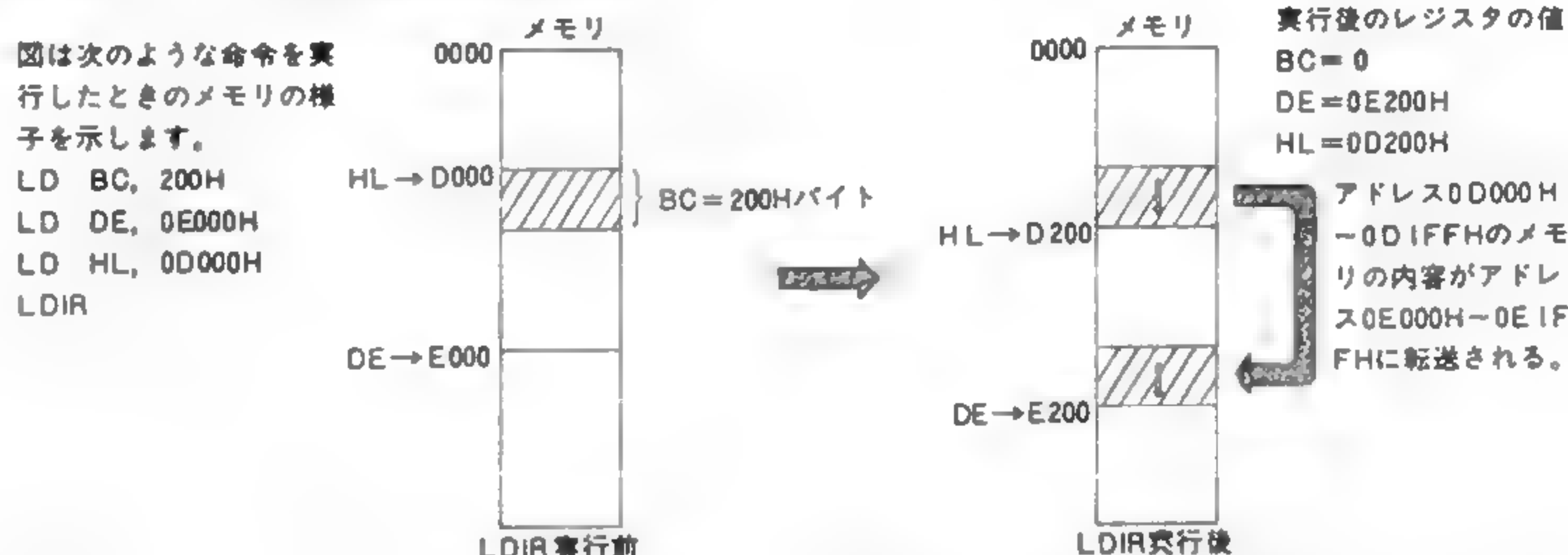


図3-12 LDDR命令の動作

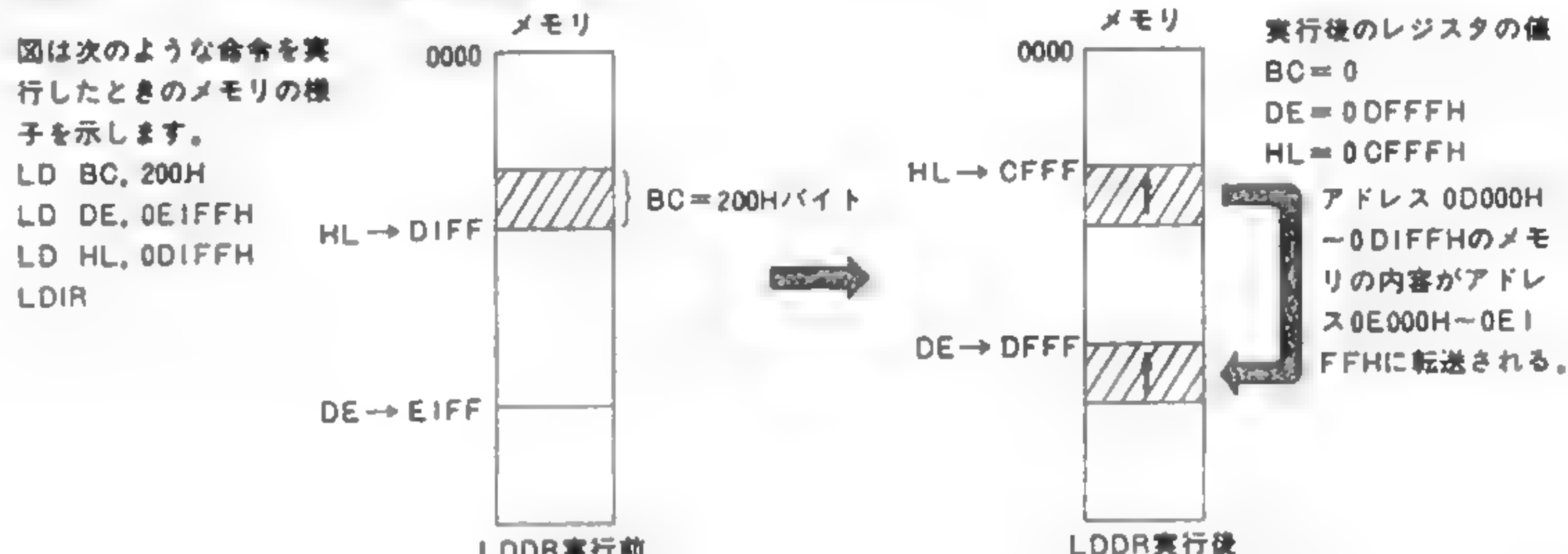


図3-13 LDIR命令しか使用できない場合

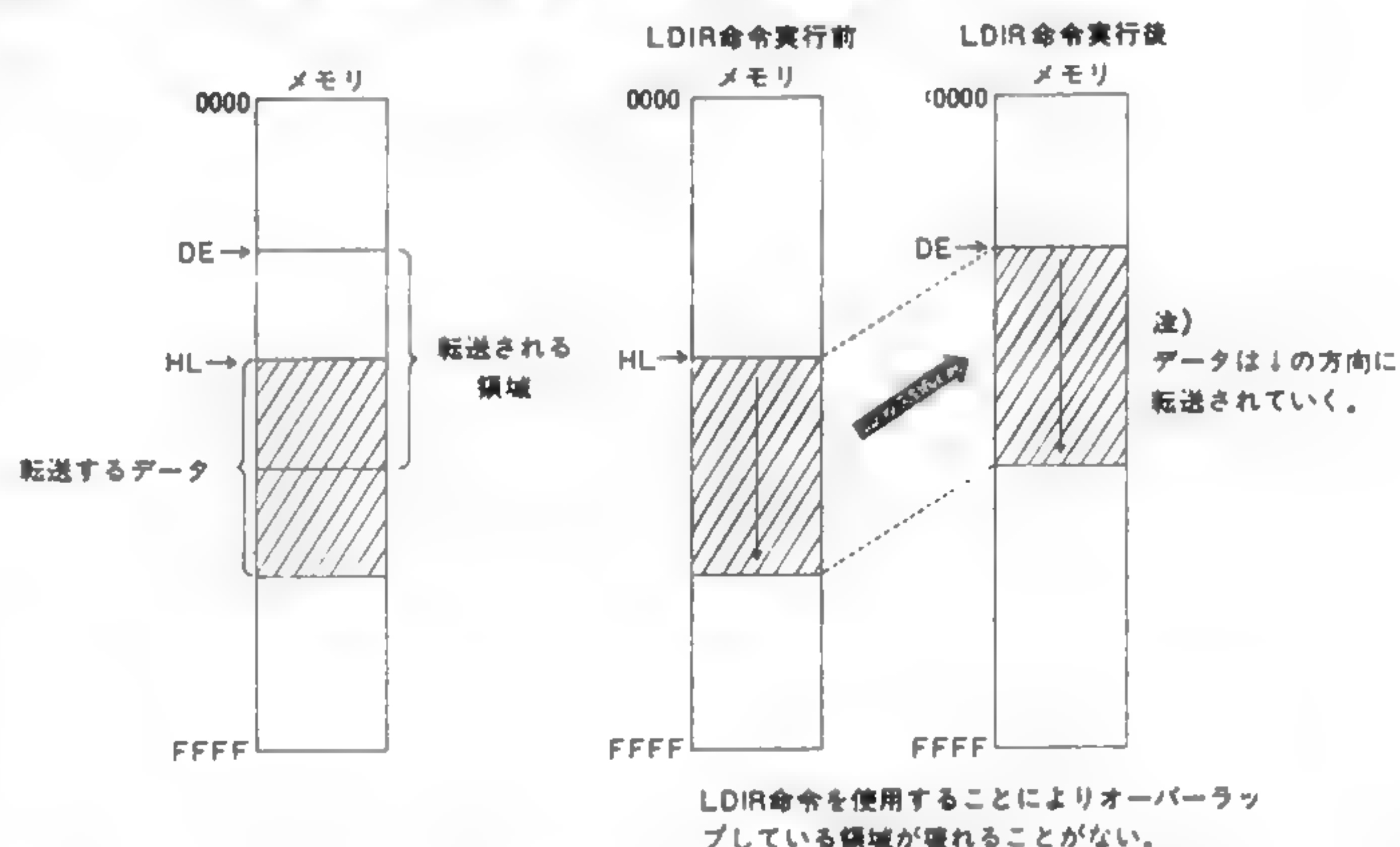
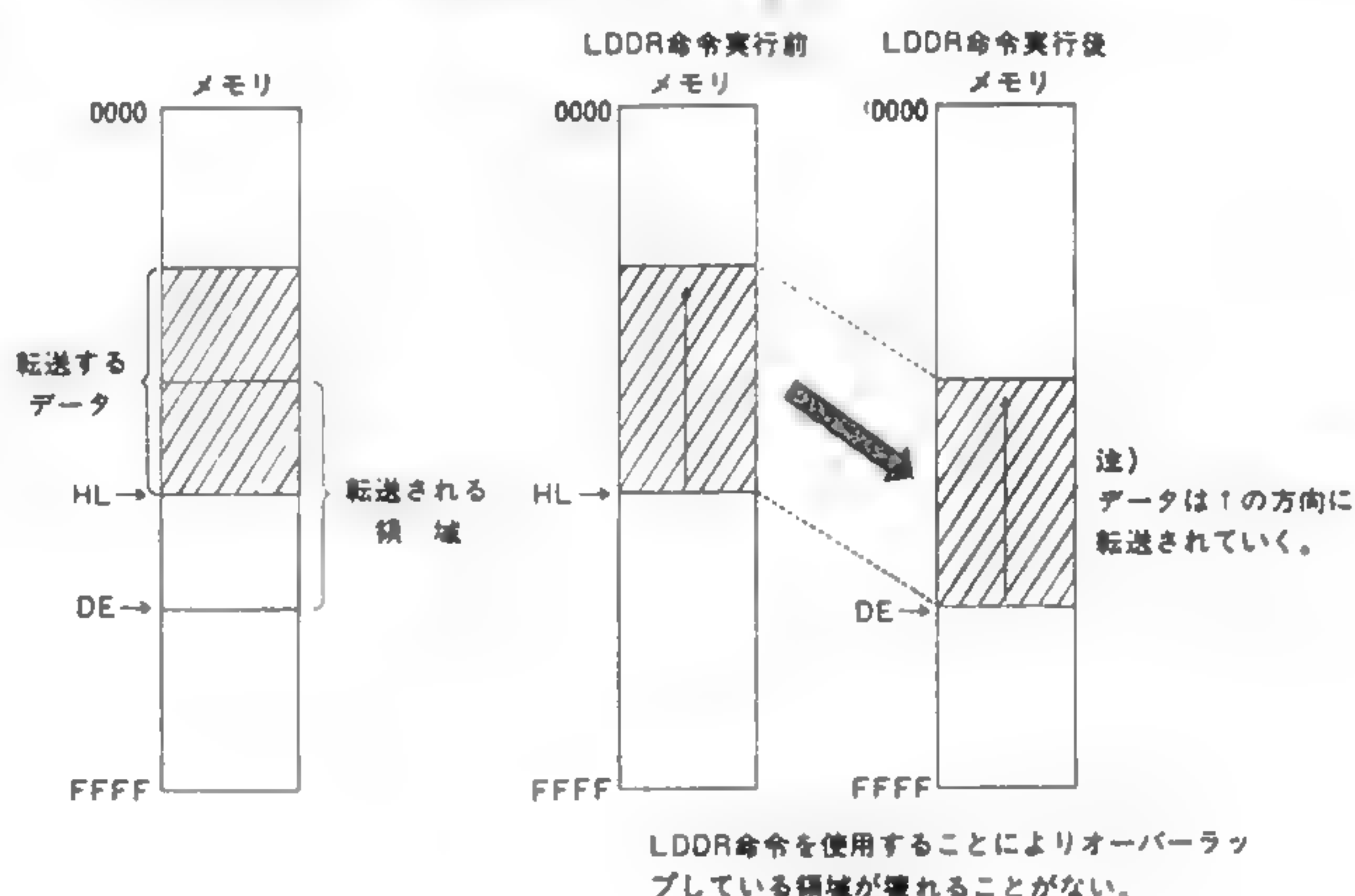


図3-14 LDDR命令しか使用できない場合



④LDIR命令, LDDR命令の転送時間

転送時間は次のように求めます。ただし、
ここで表わす式はMSXの場合です。

・BCの値が1のとき

$$\text{転送時間} = 18 \times 0.279 = 5.022 \mu\text{秒}$$

・BCの値が2以上のとき

$$\text{転送時間} = ((BC - 1) \times 23 + 18) \times 0.279 \mu\text{秒}$$

たとえば1024バイト転送するとしますと、

$$\begin{aligned} \text{転送時間} &= ((1024 - 1) \times 23 + 18) \\ &\quad \times 0.279 \mu\text{秒} \\ &= 6569.613 (\mu\text{秒}) \end{aligned}$$

かかることになります。

⑤LDI命令

LDI命令は自動で繰り返しをしない以外はLDIR命令と同じ動作をします。つまり、HLが示すメモリからデータを読み出し、DEが示すメモリに転送します。データ転送後HLとDEはインクリメント(+1)され、BCをデクリメント(-1)する、という動作を1回だけするわけです。このときBC≠0ならP/Vフラグがセットされます。

⑥LDD命令

LDD命令も自動で繰り返しをしない以外はLDDR命令と同じ動作をします。つまり、HLが示すメモリからデータを読み出し、DEが示すメモリに転送します。データ転送後HL, DE, BCをデクリメント(-1)する、という動作を1回だけするわけです。このときBC≠0ならP/Vフラグがセットされます。

2 ブロック・サーチ命令

ブロック・サーチ命令は次のペア・レジスタを特定の目的で使います。

BC: バイト数カウンタ

HL: 検索されるデータの格納アドレス

これらのペア・レジスタの内容は、ブロック転送と同様に前もって設定しておきます。

表3-12にマシン・コード、マシン・サイクル数、ステート数を示し、表3-13にブロック・サーチ命令を実行したときのフラグの変化を示します。

表3-12 ブロック・サーチ命令(CPIR, CPDR, CPI, CPD)

	マシン コード	マシン サイクル	ステート	動 作
"CPIR"	ED B1	BC≠0 and A≠(HL) 5	BC≠0 and A≠(HL) 21 (23)	Compare A:(HL), Inc HL, Dec BC repeat until BC=0 or find match
"CPDR"	ED B9	BC=0 or A=(HL) 4	BC=0 or A=(HL) 16 (18)	Compare A:(HL), Dec HL & BC repeat until BC=0 or find match
"CPI"	ED A1	4	16 (18)	Compare A:(HL) Inc HL, Dec BC
"CPD"	ED A9			Compare A:(HL) Dec HL & BC

注) ステートのカッコ内の数はMSXの場合のステート数です。

表3-13 ブロック・サーチ命令によるフラグの変化

命 令	フラグ・レジスタ							
	S	Z	H	P/V	N	CY		
CPIR;CPDR; CPI;CPD	↑	↑	×	↑	×	↑	I	•

注)

• = 実行結果の影響を受けない。

I = セットされる。

× = 不定

↑ = 実行結果の影響を受ける。

↑
もしBC-1=0ならば
P/V=0, その他P/V=1

↑
もしA=(HL)ならば
Z=1, その他Z=0

①CPIR

CPIR命令を実行すると、アキュムレータ(A)とHLが示すメモリの内容とを比較し、一致していたらZフラグをセットします。次にHLはインクリメント(+1)されBCはデクリメント(-1)されます。BCの値がゼロになるかZフラグがセットされる(一致する)までこの動作を繰り返します。CPIR命令実行後、一致するデータが見つかったらZフラグがセットされています。そしてHLは一致したデータがあるアドレスの次のアド

レスを示しています。もし一致するデータがなかったら Z フラグがリセットされています。

図3-15に CPIR 命令の動作を示します。

② CPDR

CPDR 命令は比較の後の HL のインクリメントがデクリメントになるだけで、比較、繰り返し動作とも CPIR 命令と同様です。

図3-16に CPDR 命令の動作を示します。

③ CPI

CPI 命令は繰り返しをしない以外は CPIR 命令と同じ動作をします。つまり、アキュ

ムレータ(A)とHLが示すメモリの内容とを比較し、一致していたら Z フラグをセットします。次にHLはインクリメントされBCはデクリメントされます。この動作を1回だけ行ないます。このとき $BC \neq 0$ なら P/V フラグがセットされます。

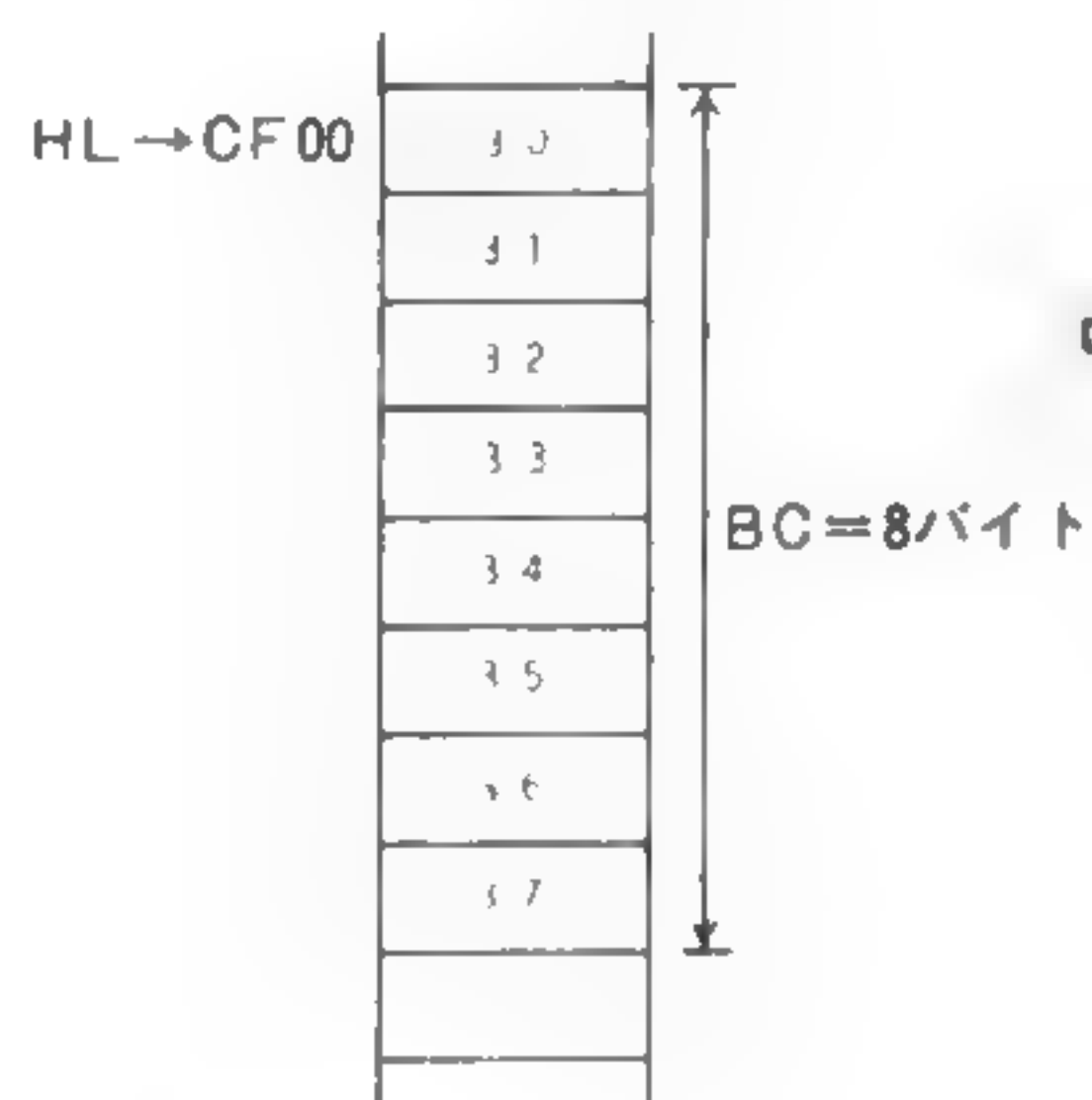
④ CPD

CPD 命令は、比較の後のHLのインクリメントがデクリメントになるだけで、CPI 命令と同じ動作をします。

図 3-15 CPIR 命令の動作

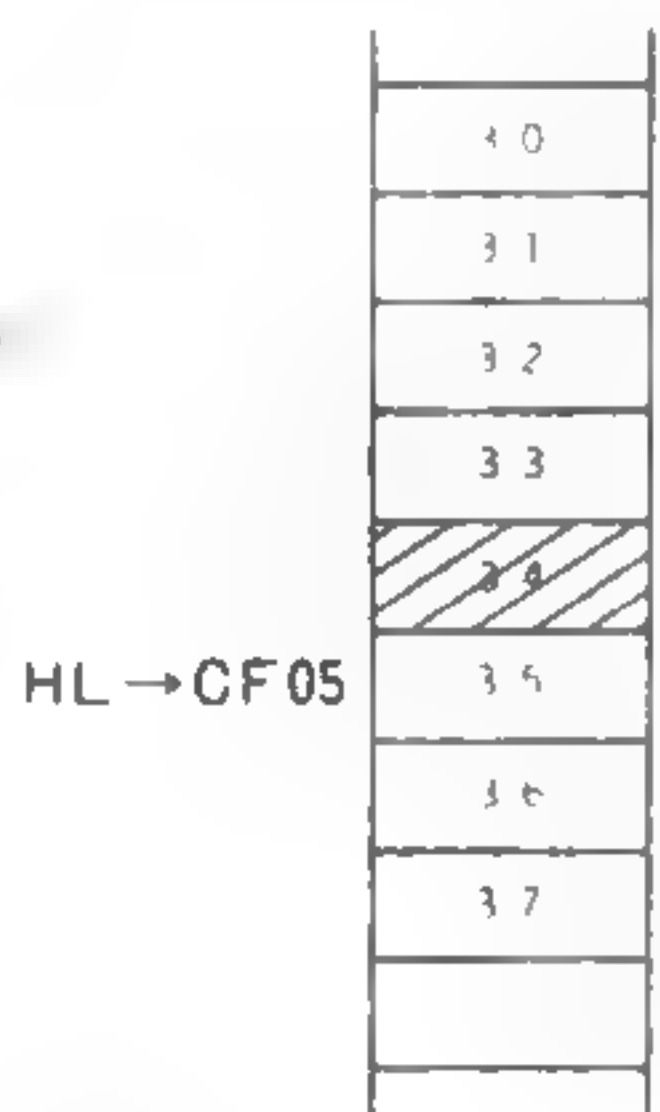
①一致するデータがある場合。

実行前のレジスタの値
A=34H BC=8
HL=0CF00H



CPIR 実行前

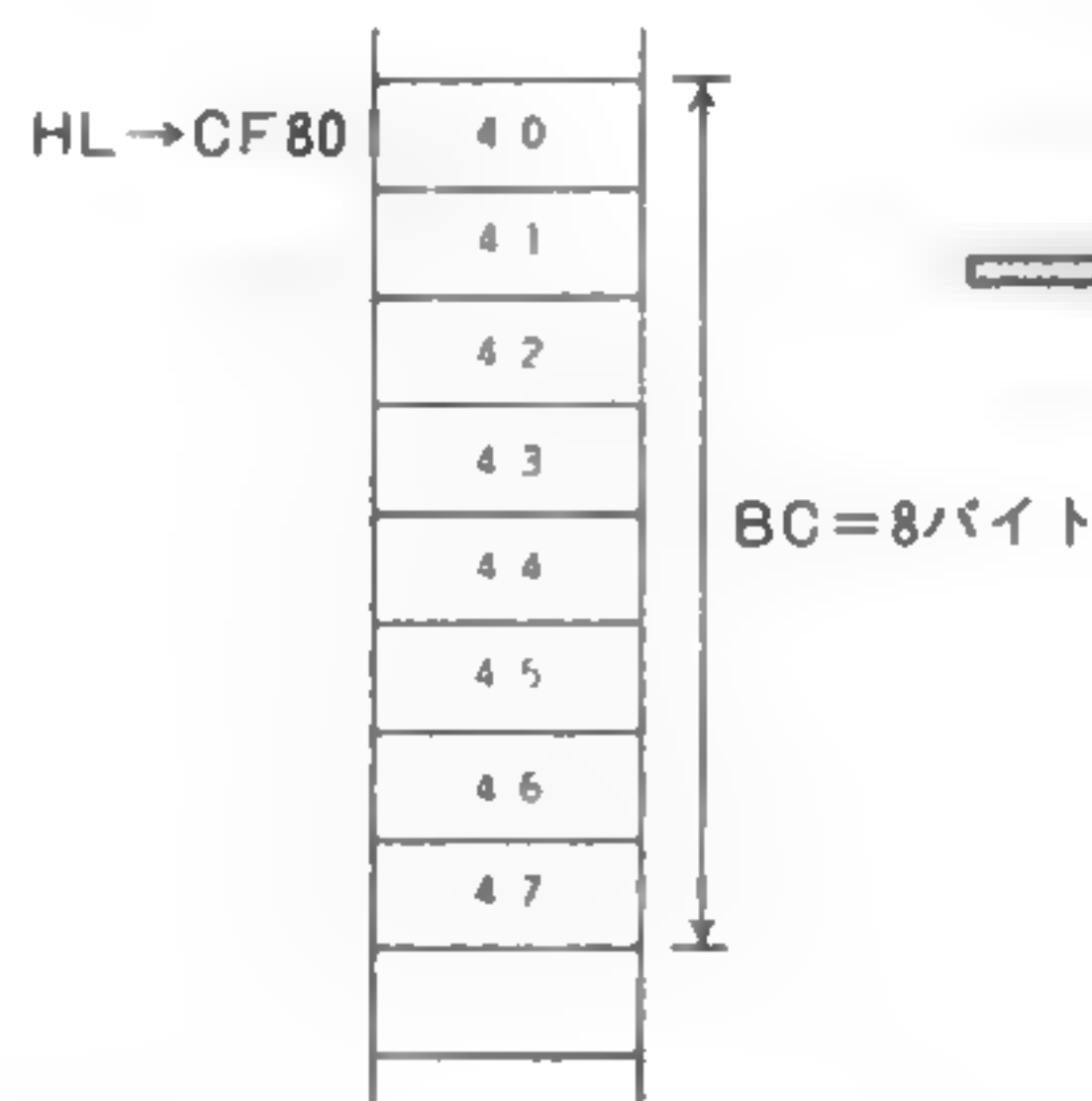
実行後のレジスタ、フラグの値
A=34H BC=3
HL=0CF05H
Zフラグ=1, P/Vフラグ=1



CPIR 実行後

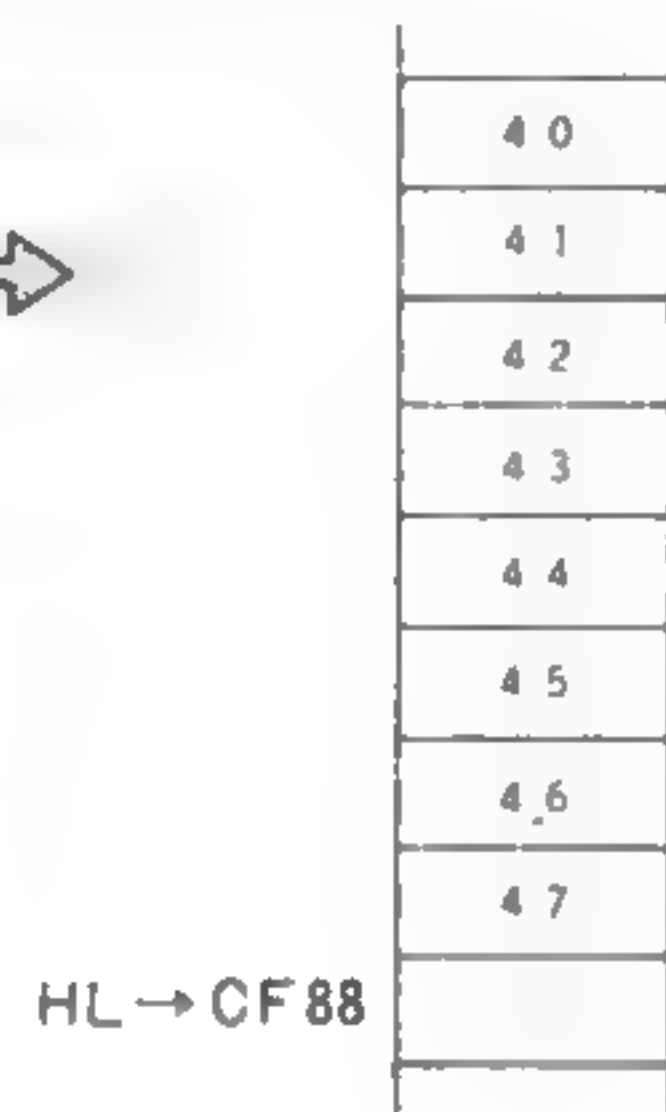
②一致するデータがない場合。

実行前のレジスタの値
A=34H BC=8
HL=0CF80H



CPIR 実行前

実行後のレジスタ、フラグの値
A=34H BC=0
HL=0CF88H
Zフラグ=0, P/Vフラグ=0

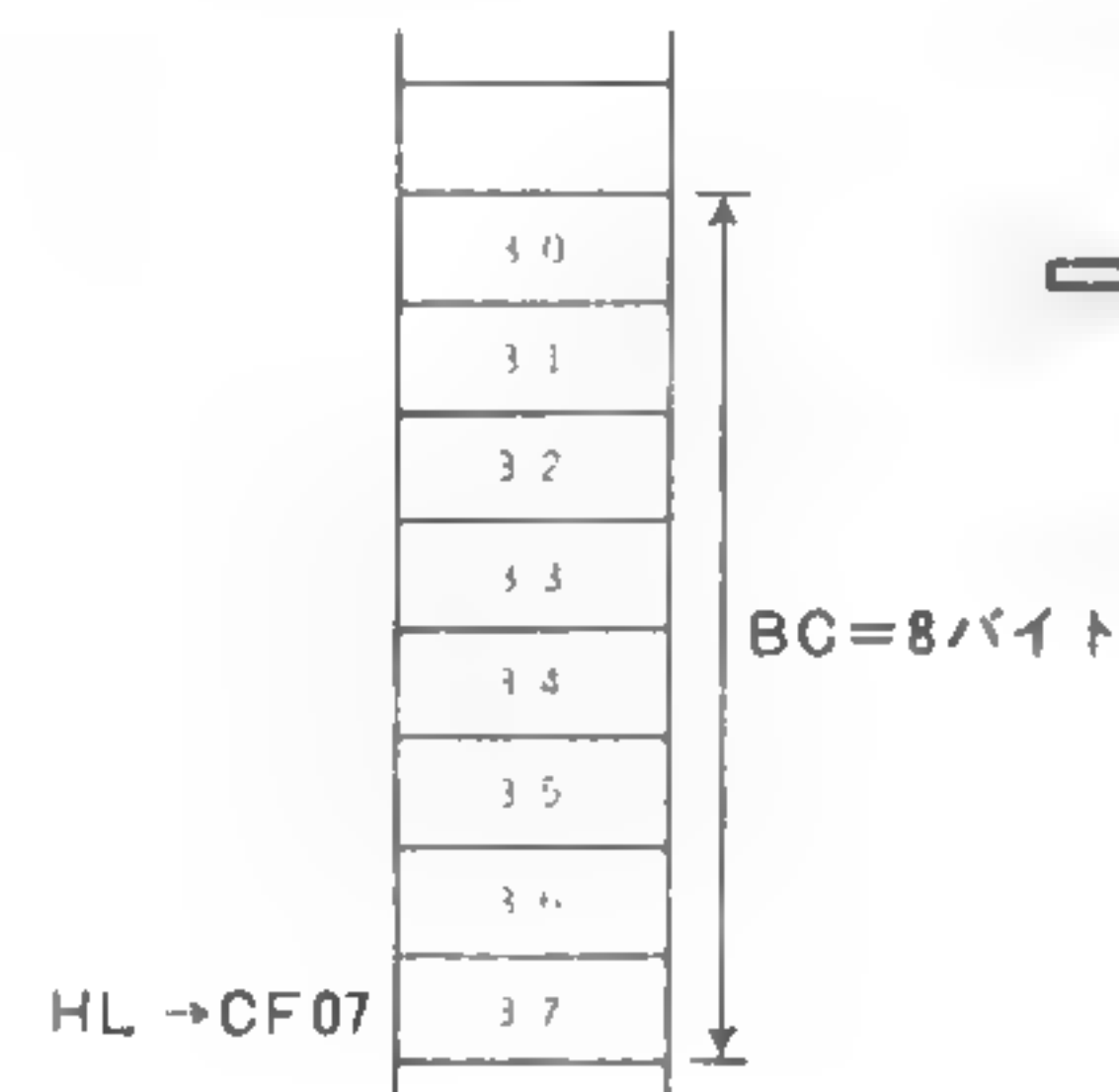


CPIR 実行後

図 3-16 CPDR 命令の動作

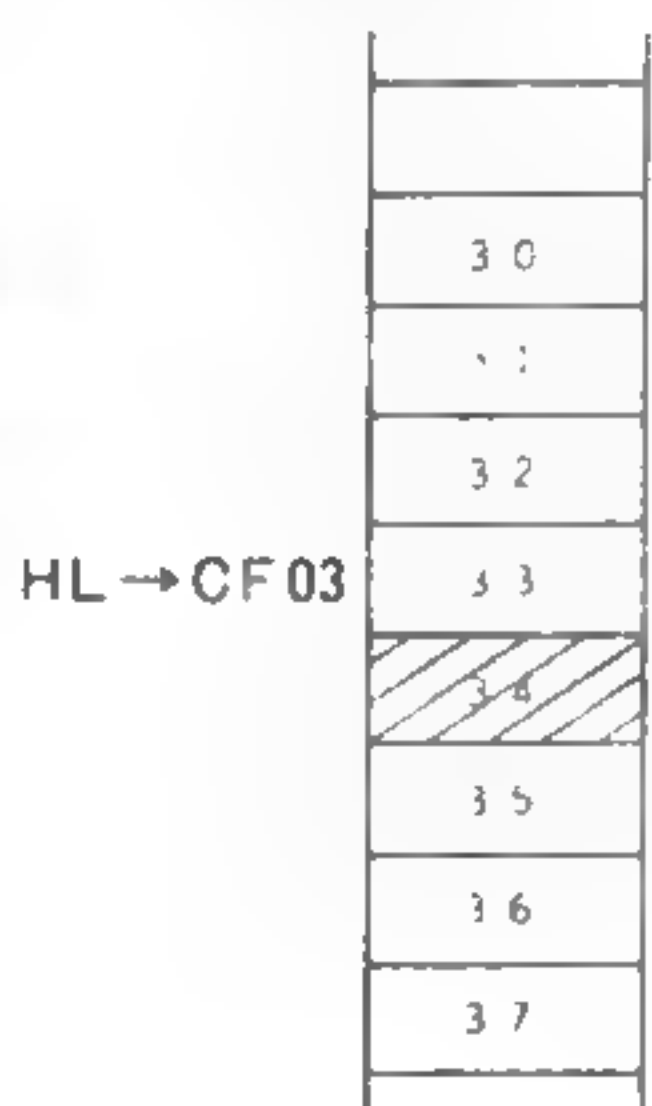
①一致するデータがある場合。

実行前のレジスタの値
A=34H BC=8
HL=0CF07H



CPDR 実行前

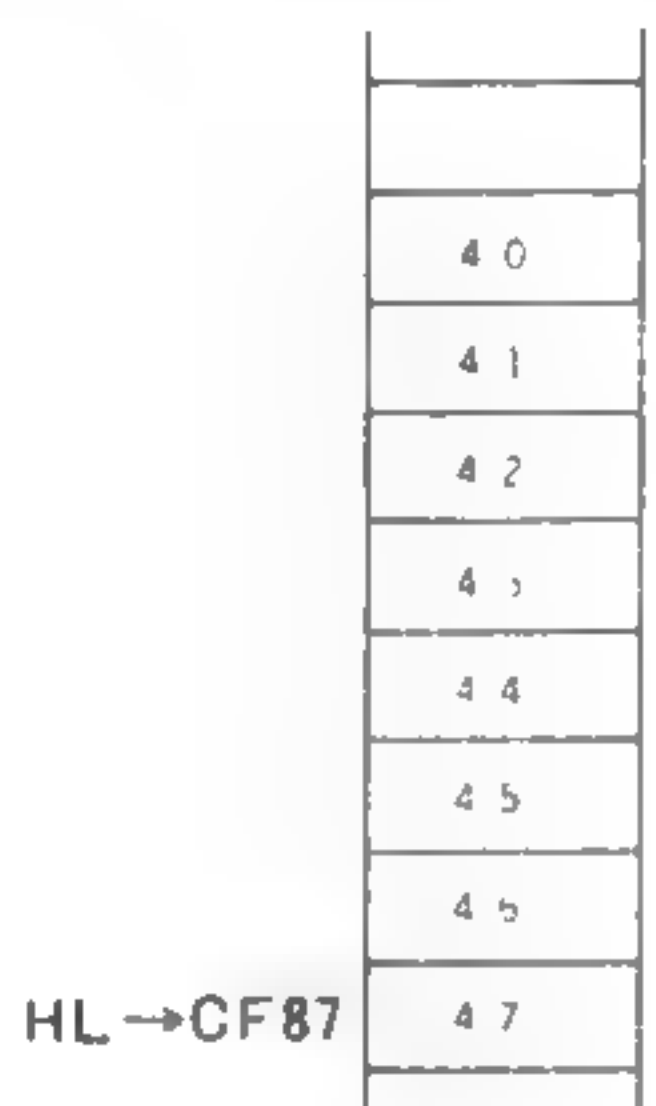
実行後のレジスタ、フラグの値
A=34H BC=4
HL=0CF03H
Zフラグ=1, P/Vフラグ=1



CPDR 実行後

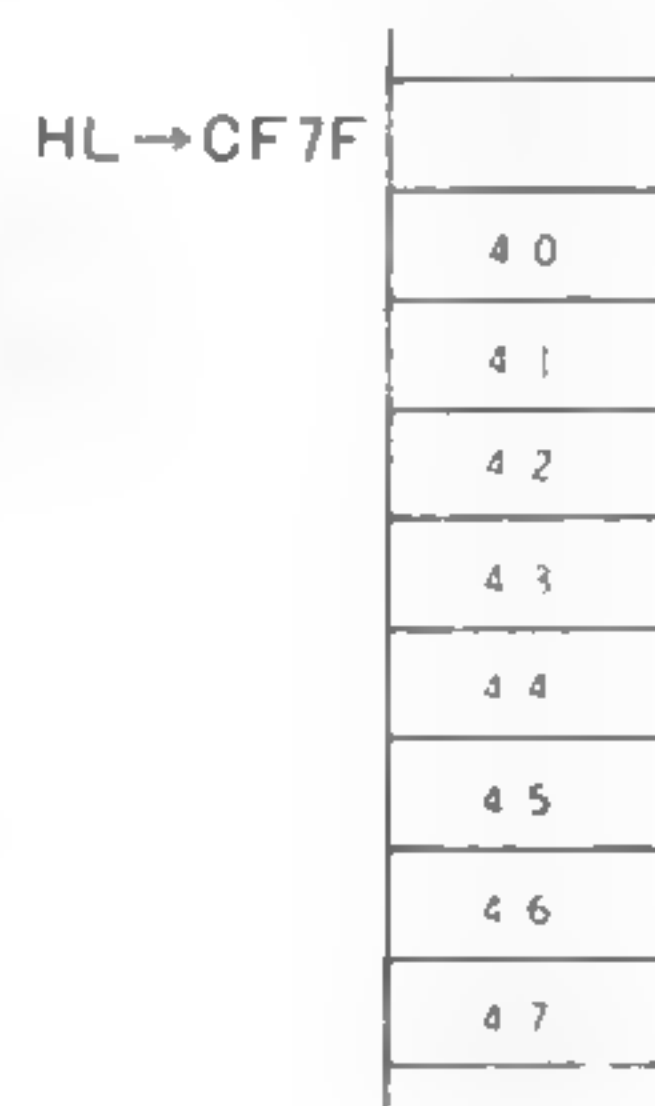
②一致するデータがない場合。

実行前のレジスタの値
A=34H BC=8
HL=0CF87H



CPDR 実行前

実行後のレジスタ、フラグの値
A=34H BC=0
HL=0CF7FH
Zフラグ=0, P/Vフラグ=0



CPDR 実行後

4. 演算命令

演算命令には8ビット演算と16ビット演算の2種類があります。

8ビット演算には、

ADD

ADC (ADd with Carry)

SUB (SUBtract)

SBC (SuBtract with Carry)

AND

XOR (eXclusive OR)

OR

CP (ComPare)

INC (INCrement)

DEC (DECrement)

DAA (Decimal Adjust Accumulator)

CPL (ComPLement accumulator)

NEG (NEGate accumulator)

の13個の命令があります。

16ビットの演算命令は算術演算だけで、

ADD

ADC (ADd with Carry)

SBC (SuBtract with Carry)

INC (INCrement)

DEC (DECrement)

の5つの命令があります。

① 8ビット演算命令

8ビット演算のマシン・コードを表3-14に示します。この中でINC、DECを除いた演算命令は、アキュムレータAの内容と表の上欄に示すレジスタ/メモリのデータとの間で演算を行ない、その結果をアキュムレータAに記憶して結果の状態をフラグ・レジスタに設定します。ただし、CP命令は演算した後、結果をアキュムレータAに記憶しません。つまり、CP命令を実行してもアキュムレータAの内容は変化しません。

表3-14 8ビット算術・論理演算命令

①オペランドがある8ビット演算命令

オペランド 命令	REGISTER							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
ADD "ADD A, opr"	87	80	81	82	83	84	85	86	$\begin{smallmatrix} DD \\ 86 \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ 86 \\ d \end{smallmatrix}$	$\begin{smallmatrix} C6 \\ n \end{smallmatrix}$
ADD WITH CARRY "ADC A, opr"	8F	88	89	8A	8B	8C	8D	8E	$\begin{smallmatrix} DD \\ 8E \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ 8E \\ d \end{smallmatrix}$	$\begin{smallmatrix} CE \\ n \end{smallmatrix}$
SUBTRACT "SUB opr"	97	90	91	92	93	94	95	96	$\begin{smallmatrix} DD \\ 96 \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ 96 \\ d \end{smallmatrix}$	$\begin{smallmatrix} D6 \\ n \end{smallmatrix}$
SUB WITH CARRY "SBC A, opr"	9F	98	99	9A	9B	9C	9D	9E	$\begin{smallmatrix} DD \\ 9E \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ 9E \\ d \end{smallmatrix}$	$\begin{smallmatrix} DE \\ n \end{smallmatrix}$
AND "AND opr"	A7	A0	A1	A2	A3	A4	A5	A6	$\begin{smallmatrix} DD \\ A6 \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ A6 \\ d \end{smallmatrix}$	$\begin{smallmatrix} E6 \\ n \end{smallmatrix}$
EXCLUSIVE OR "XOR opr"	AF	A8	A9	AA	AB	AC	AD	AE	$\begin{smallmatrix} DD \\ AE \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ AE \\ d \end{smallmatrix}$	$\begin{smallmatrix} EE \\ n \end{smallmatrix}$
OR "O R opr"	B7	B0	B1	B2	B3	B4	B5	B6	$\begin{smallmatrix} DD \\ B6 \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ B6 \\ d \end{smallmatrix}$	$\begin{smallmatrix} F6 \\ n \end{smallmatrix}$
COMPARE "C P opr"	BF	B8	B9	BA	BB	BC	BD	BE	$\begin{smallmatrix} DD \\ BE \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ BE \\ d \end{smallmatrix}$	$\begin{smallmatrix} FE \\ n \end{smallmatrix}$
INCREMENT "INC opr"	3C	04	0C	14	1C	24	2C	34	$\begin{smallmatrix} DD \\ 34 \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ 34 \\ d \end{smallmatrix}$	
DECREMENT "DEC opr"	3D	05	0D	15	1D	25	2D	35	$\begin{smallmatrix} DD \\ 35 \\ d \end{smallmatrix}$	$\begin{smallmatrix} FD \\ 35 \\ d \end{smallmatrix}$	

②オペランドがない8ビット演算命令

Decimal Adjust Acc "D A A"	27
Complement Acc "C P L"	2F
Negate Acc "NEG" (2's complement)	ED 44

注)

- ADD, ADC, SBC 命令は2つのオペランドがあり、第1オペランドは"A"、第2オペランドの方で表を引く。他の命令はオペランドが1つ。
- dはディスプレイメント(2の補数で-128~+127)
- nは1バイトの値

INC, DEC 命令は表の上欄に示すレジスタ/メモリの内容を INC 命令で +1, DEC 命令で -1 します。

表3-15に8ビット演算命令のマシン・サイクル数, ステート数を, 表3-16に8ビット演算命令実行によるフラグの変化を示します。

表3-15 8ビット演算命令のサイクル数とステート数

①オペランドがある8ビット演算命令

命令	REGISTER							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
ADD A, opr	マシン・サイクル M=1 ステート T=4(5)							M=2 T=7(8)	M=5 T=19(21)		M=2 T=7(8)
ADC A, opr											
SUB opr											
SBC A, opr											
AND opr											
XOR opr											
OR opr											
CP opr											
INC opr	M=1							M=3 T=11(12)	M=6 T=23(25)		
DEC opr	T=4(5)										

注)

- ・ Mはマシン・サイクル
- ・ Tはステートのこと。
- ・ カッコ内の数は MSX の場合のステート数。

②オペランドがない8ビット演算命令

DAA	M=1, T=4(5)
CPL	M=1, T=4(5)
NEG	M=2, T=8(10)

表3-16 8ビット演算命令によるフラグの変化

命令	S	Z	H	P _V	N	CY
ADD A, s; ADC A, s	⋮	⋮	X	⋮	X	V 0
SUBs; SBC A, s; CPs; NEG	⋮	⋮	X	⋮	X	V 1
ANDs	⋮	⋮	X	1	X	P 0 0
ORs; XORs	⋮	⋮	X	0	X	P 0 0
INCs	⋮	⋮	X	⋮	X	V 0
DECs	⋮	⋮	X	⋮	X	V 1
DAA	⋮	⋮	X	⋮	X	P
CPL	⋮	⋮	X	1	X	⋮

- 注) ・ = 実行結果の影響を受けない 0 = リセットされる
 1 = セットされる X = 不定 ⋮ = 実行結果の影響を受ける
 ・ 命令中の S はオペランド。
 ・ P_V フラグ
 { 論理演算の場合: 偶数パリティ → P=1, 奇数パリティ → P=0
 算術演算の場合: オーバーフローあり → V=1,
 オーバーフローなし → V=0

① ADD 命令

ADD 命令はアキュムレータ A の内容にレジスタ/メモリの内容を 8 ビットで加算し, 結果をアキュムレータに格納します。

たとえば, A=3AH, E=29H のとき「ADD A, E」を実行すると A=63H になります。

アキュムレータ A=3AH 00111010
 レジスタ E =29H +00101001
 加算結果 A=63H 01100011

ADD 命令実行後のフラグの状態は,

CY: 結果が 8 ビットを超えたときセットされます (ビット 7 からのキャリーが入る)。

N: リセットされます。

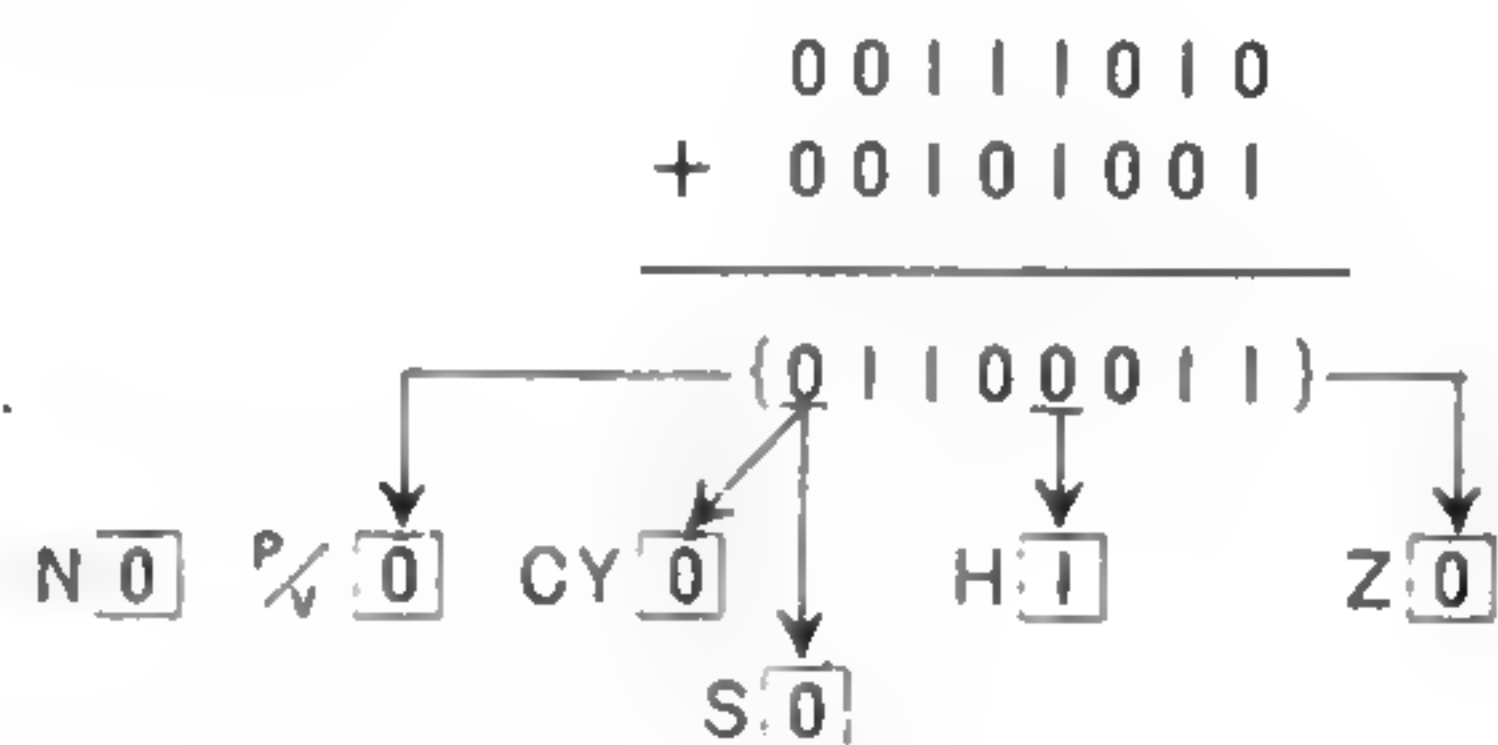
P/V: 結果が 2 の補数の範囲 (10 進数で -128 ~ +127) を超えたときセットされます。

H: ビット 3 からのキャリーが入ります。

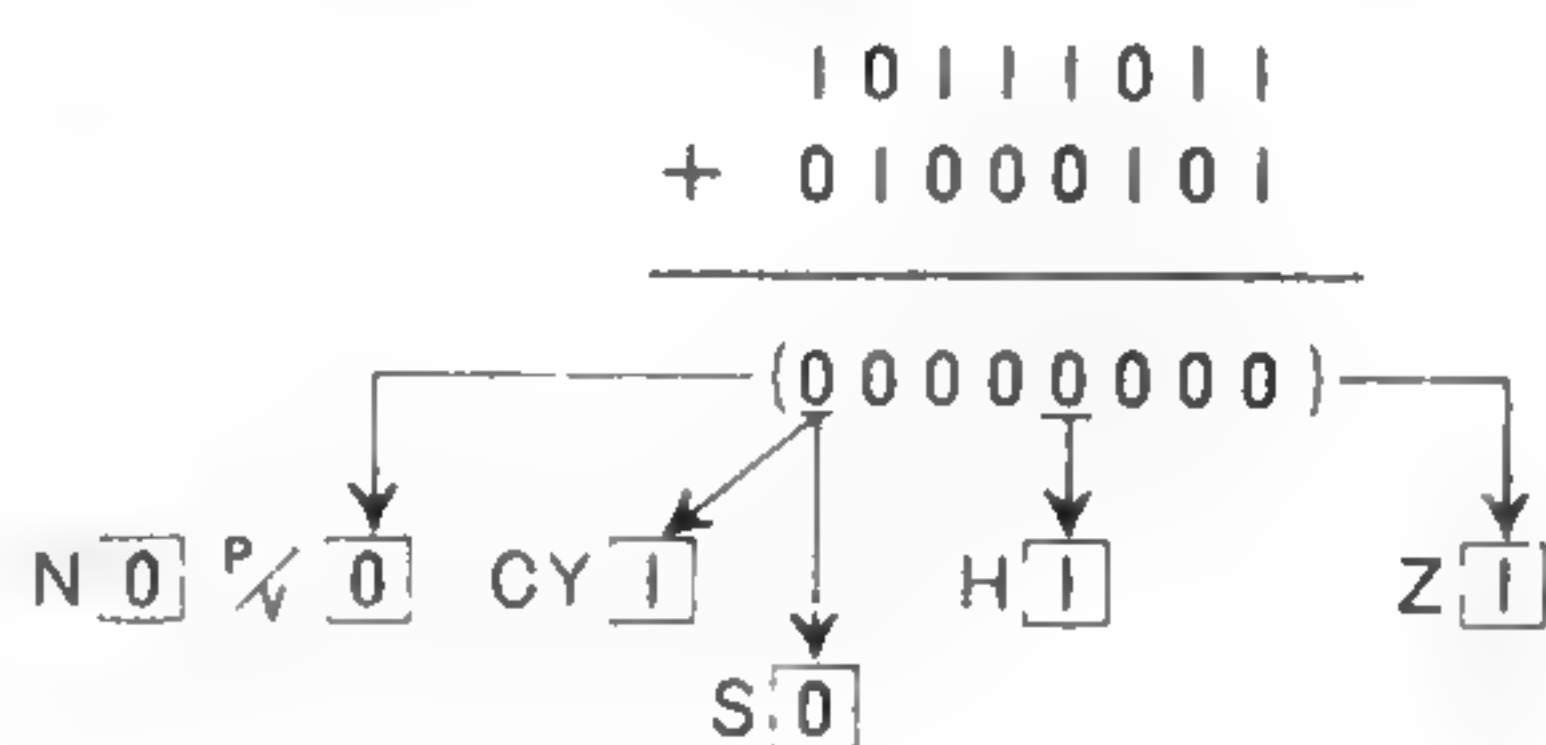
Z: 加算後アキュムレータの全ビットがゼロのときセットされます。

S: 結果が 2 の補数の値として負 (80H ~ 0FH) になったときセットされます。

たとえば 3AH+29H を計算するとフラグは次のようになります。



次の例は 0BBH+45H を計算したときです。



②ADC命令

ADC命令はキャリーフラグの内容をも含めて加算します。

たとえばA=3AH, E=29Hのとき「ADC A, E」を実行すると、実行前のキャリーフラグが0ならA=63Hになり、キャリーフラグが1なら64Hになります。

●CY=0の場合

アキュムレータ A=3AH	00111010
レジスタ E =29H	+00101001
キャリー フラグ =0	+ 0
加算結果	01100011

アキュムレータ A=63H

●CY=1の場合

アキュムレータ A=3AH	00111010
レジスタ E =29H	+00101001
キャリー フラグ =1	+ 1
加算結果	01100100

アキュムレータ A=64H

ADC命令実行後のフラグの状態はADD命令と同じです。

③SUB命令

SUB命令はアキュムレータAの内容からレジスタ/メモリの内容を8ビットで減算し、結果をアキュムレータに格納します。

たとえば、A=0E9H, HLペア・レジスタが示すアドレスのメモリの内容が53Hのとき、「SUB(HL)」を実行するとA=96Hになります。

アキュムレータ A=0E9H	11101001
(HL) =53H	-01010011
減算結果 A=96H	10010110

SUB命令実行後のフラグの状態は、

CY:符号なし整数として見たとき、引かれる数より引く数の方が大きいときセットさ

れます(ビット7からのボローが入る)。

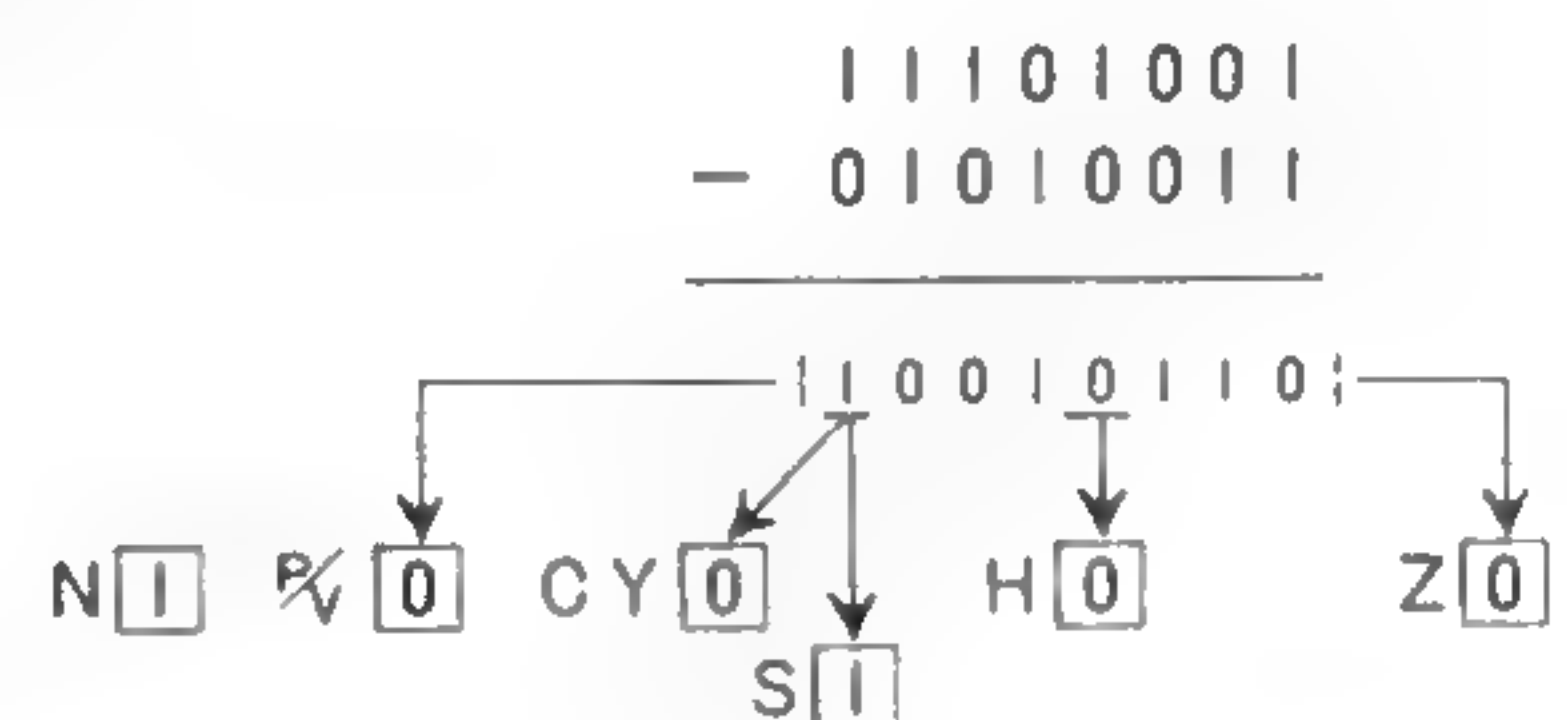
N:セットされます。

P/V:結果が2の補数の範囲(10進数で-128~+127)を超えた場合セットされます。

H:ビット3の減算時ボローが発生すればセットされます。

S:結果が2の補数の値として負(80H~0FFH)になった場合セットされます。

たとえば0E9H-53Hを計算するとフラグは次のようになります。



④SBC命令

SBC命令はキャリーフラグの内容(下位からのボロー)も含めて減算する命令です。

たとえば、A=0E9H, HLが示すアドレスのメモリの内容が53Hのとき「SBC A, (HL)」を実行すると、実行前のキャリーフラグが0ならA=96Hとなり、キャリーフラグが1ならA=95Hになります。

●CY=0の場合

アキュムレータ A=0E9H	11101001
(HL) =53H	-01010011
キャリー フラグ=0	- 0
減算結果	10010110

アキュムレータ A=96H

●CY=1の場合

アキュムレータ A=0E9H	11101001
(HL) =53H	-01010011
キャリー フラグ=1	- 1
減算結果	10010101

アキュムレータ A=95H

SBC命令実行後のフラグの状態はSUB命令と同じです。

⑤AND, OR, XOR命令

AND, OR, XOR命令はアキュムレータAの内容とレジスタ/メモリの内容を1ビットずつそれぞれ論理演算をして、結果をアキュムレータに格納します。

たとえば、A=3AH, C=0A3Hのとき「AND C」を実行するとA=22Hになり、「OR C」を実行するとA=0BBHに、「XOR C」ならA=99Hとなります。

●AND C

アキュムレータA=3AH 00111010
 レジスタC =0A3H AND 10100011
 演算結果 00100010
 アキュムレータA=22H

●OR C

アキュムレータA=3AH 00111010
 レジスタC =0A3H OR 10100011
 演算結果 10111011
 アキュムレータA=0BBH

●XOR C

アキュムレータA=3AH 00111010
 レジスタC =0A3H XOR 10100011
 演算結果 10011001
 アキュムレータA=99H

AND, OR, XOR各命令実行後のフラグの変化はHフラグを除いて同じです。
 CY:リセットされます。
 N:リセットされます。
 P/V:演算後アキュムレータAの8ビットのうち、1になっているビットの数が偶数ならセットされます。
 H:AND命令ではセットされ、OR/XOR命令ではリセットされます。

Z:演算結果が00Hならセットされます。
 S:結果が2の補数の値として負(80H~0FFH)ならセットされます。

AND, OR, XOR命令は、通常の論理演算に使われるほかにアキュムレータAをオペランドにすることで次のような操作ができます。

i)「AND A」または「OR A」

アキュムレータAの値を変化させずにCYフラグのリセットができるほかアキュムレータAの状態調べられます(A=00Hの場合はZフラグがセットされ、A=80H~0FFHの場合はSフラグがセットされる)。

ii)「XOR A」

アキュムレータAを00Hにクリアします。

⑥CP命令

CP命令はSUB命令と同じようにアキュムレータAの内容からレジスタ/メモリの内容を8ビットで減算しますが、減算の結果は使われません。つまり、アキュムレータAの値は変化しません。ただし、フラグはSUB命令と同じように変化します。

この命令は、後述する条件ジャンプ、条件コール、条件リターン命令と組み合わせて使います。

⑦INC命令

レジスタ/メモリの内容に1を加える命令です。たとえばL=3FHのとき「INC L」を実行するとL=40Hになります。

INC命令実行後のフラグの変化は、
 CY:変化しません。
 N:リセットされます。
 P/V:結果が80Hになったときセットされます。
 H:ビット3からのキャリーが入ります。

Z:結果が00Hになったときセットされます。

S:結果が2の補数の値として負(80H~0FFH)になったときセットされます。

INC命令は、ADD命令とは異なりキャリーフラグを変化させることはありません。たとえばA=0FFHのとき「ADD A, 1」を実行するとA=00H, CY=1となりますが、「INC A」を実行するとA=00Hにはなってもキャリーフラグは元のままで変化しません。

⑧DEC命令

レジスタ/メモリの内容から1を引く命令です。たとえばH=60Hのとき「DEC H」を実行するとH=5FHになります。

DEC命令実行後のフラグの変化は、

CY:変化しません。

N:セットされます。

P/V:結果が7FHになったときセットされます。

H:ビット3からの borrowが入ります。

Z:結果が00Hになったときセットされます。

S:結果が2の補数の値として負(80H~0FFH)になったときセットされます。

DEC命令はINC命令と同じくキャリーフラグを変化させることはありません。

⑨DAA命令

DAA命令を加算命令(ADD, ADC)や減算命令(SUB, SBC)と組み合わせて使うことによりBCDでの10進演算ができます。

加算や減算の命令は2進数として演算しているため、演算前の値がBCDであっても演算後の値はBCDではなくなってしまいます。それを補正しBCDにするのがDAA命令です。

DAA命令はキャリーフラグ(CY)とハーフ・キャリーフラグ(H)を使って加減算後のアキュムレータの値を2桁のBCDに変換します。ハーフ・キャリーフラグはこの命令のた

めに設けられたフラグなのです。

図3-17はDAA命令の動作を示したものです。たとえばA=63H, C=28Hのときに、

ADD A, C

DAA

という命令を実行すると、A=91Hとなり

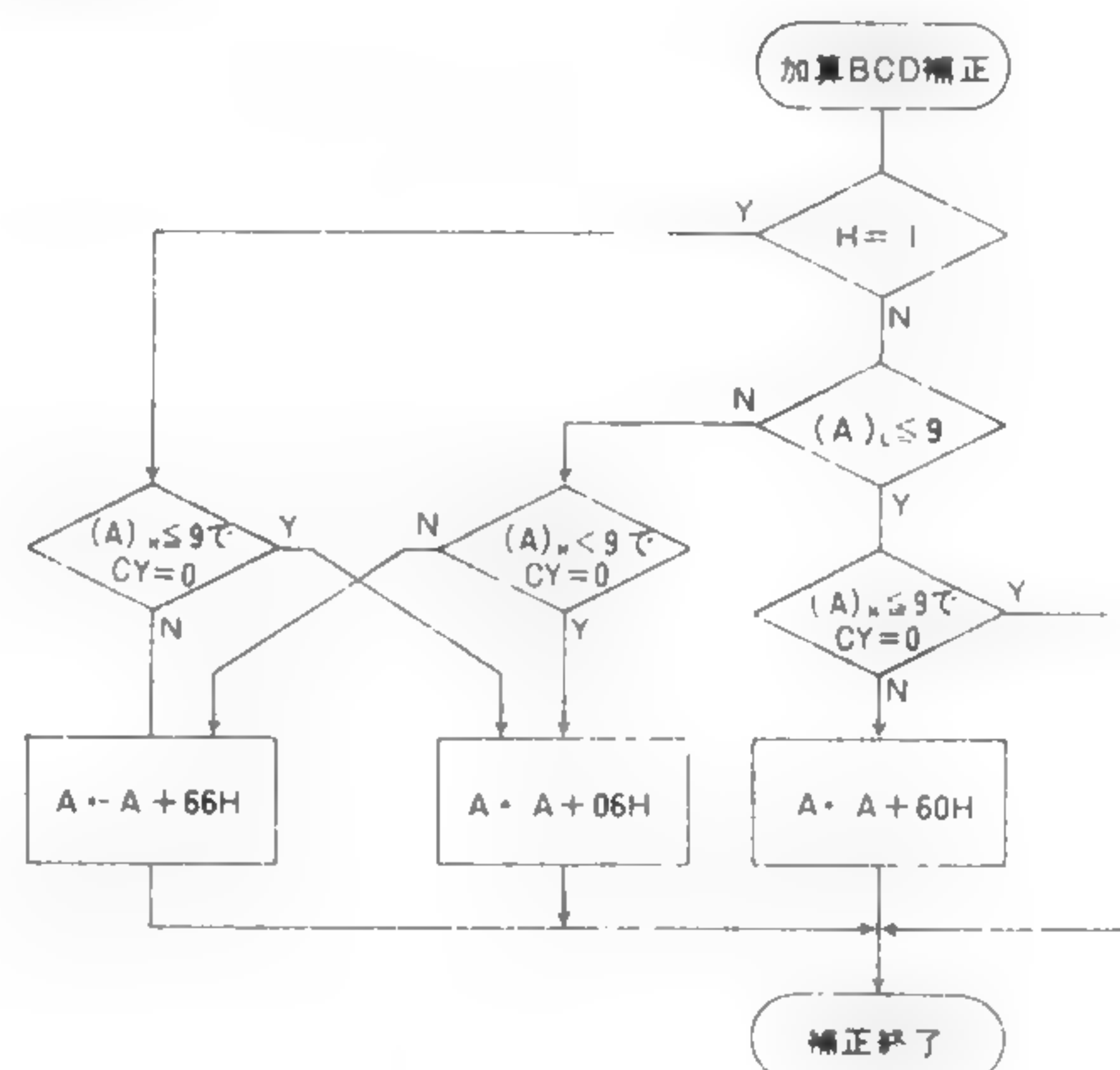
SUB A, C

DAA

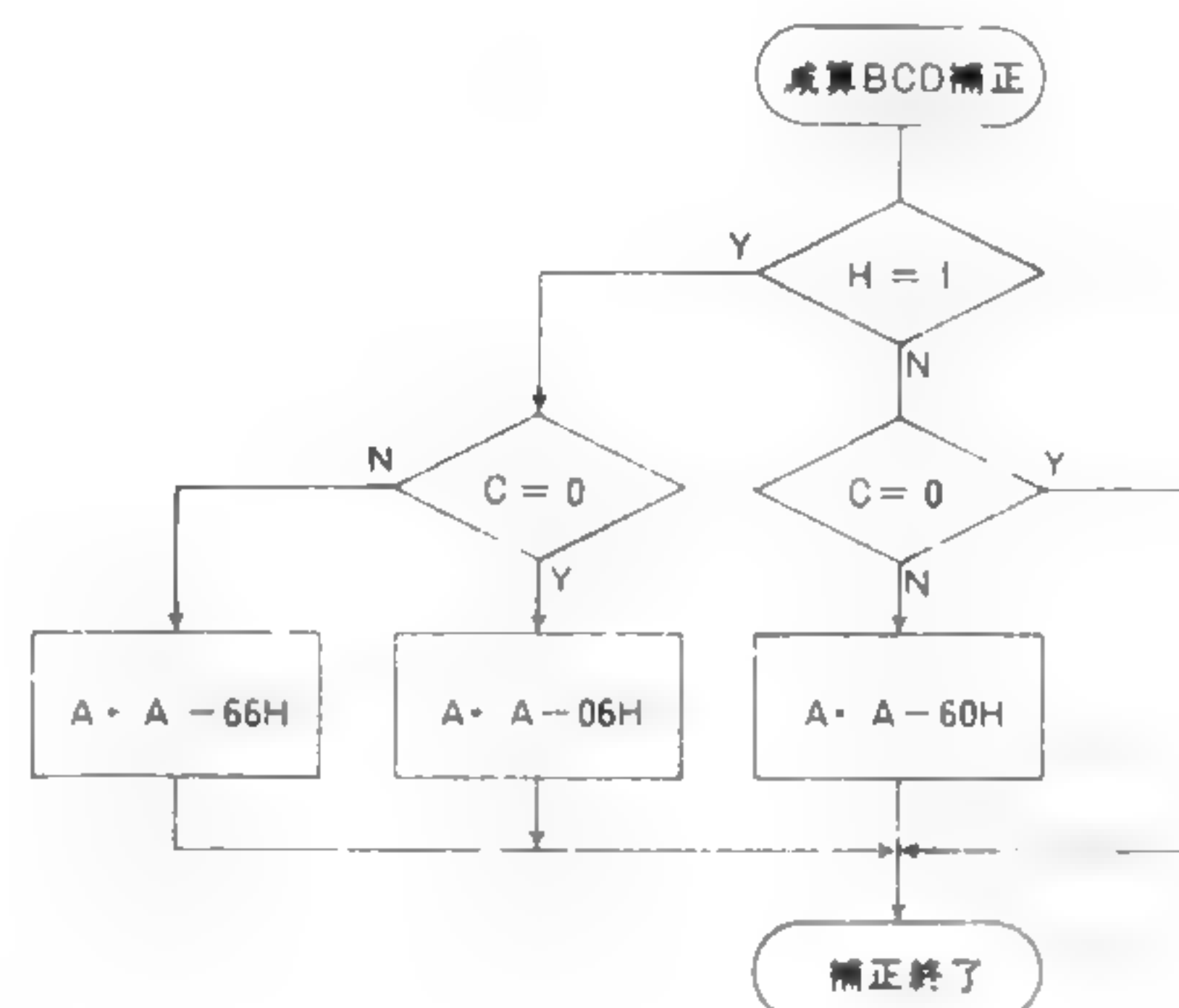
を実行するとA=35Hになります。

図3-17 DAA命令の動作

●Nフラグ=0



●Nフラグ=1



注)

- CY:キャリーフラグ
- H:ハーフ・キャリーフラグ
- A:アキュムレータA
- (A)₄:アキュムレータ上位4ビット
- (A)₂:アキュムレータ下位4ビット

D A A 命令の実行によるフラグの変化は次のようになります。

CY: 上位 4 ビットを 10 進数に変換したとき、キャリー/ボローが発生すればセットされます

(たとえば、アキュムレータで 83H + 54H を演算し D A A による補正をした後は C Y = 1, A = 37H になる)。

N: 変化しません。

P/V: 命令実行後アキュムレータ A の 8 ビットのうち、1 になっているビットの数が偶数ならセットされます。

H: 下位 4 ビットを 10 進数に変換したときキャリー/ボローが発生すればセットされます。

Z: 命令実行後アキュムレータ A が 00H になっていればセットされます。

S: 命令実行後のアキュムレータ A の最上位ビット (ビット 7) と同じになります。

⑩ C P L 命令

C P L 命令はアキュムレータの全ビットを反転させます (1 の補数をとる)。

たとえば A = 0D9H のとき C P L 命令を実行すると A = 26H となります。

C P L 命令実行前 11011001

↓

C P L 命令実行後 00100110

C P L 命令では N フラグと H フラグがセットされる以外、ほかのフラグは影響されません。

⑪ N E G 命令

N E G 命令はアキュムレータの値の 2 の補数をとります。つまり、00H からアキュムレータの内容を減算し結果をアキュムレータに格納します。

たとえば A = 4CH のとき N E G 命令を実行すると A = 0B4H になります。

N E G 命令によるフラグの変化は S U B 命令と同じです。

2 16ビット演算命令

16 ビット演算のマシン・コードを表 3-17 に示します。16 ビット算術演算命令は 16 ビット・レジスタ間で演算できるだけで、メモリ上にある 16 ビット・データと直接演算することはできません。

表 3-18 に 16 ビット演算命令のマシン・サイクル数、ステート数を、表 3-19 に 16 ビット演算命令実行によるフラグの変化を示します。

表 3-17 16 ビット算術演算命令

命 令	オペランド	REGISTER					
		BC	DE	HL	SP	IX	IY
ADD	"ADDHL, opr"	0 9	1 9	2 9	3 9		
	"ADDIX, opr"	DD 0 9	DD 1 9		DD 3 9	DD 2 9	
	"ADDIY, opr"	FD 0 9	FD 1 9		FD 3 9		FD 2 9
ADD WITH CARRY AND SET FLAGS	"ADCHL, opr"	ED 4 A	ED 5 A	ED 6 A	ED 7 A		
SUB WITH CARRY AND SET FLAGS	"SBCHL, opr"	ED 4 2	ED 5 2	ED 6 2	ED 7 2		
INCREMENT	"INC opr"	0 3	1 3	2 3	3 3	DD 2 3	FD 2 3
DECREMENT	"DEC opr"	0 B	1 B	2 B	3 B	DD 2 B	FD 2 B

注) ADD, ADC, SBC 命令は 2 つオペランドがあり、第 1 オペランドはペア・レジスタ "HL", "IX", "IY", 第 2 オペランドの方で表を引く。

INC, DEC 命令はオペランドが 1 つ。

表 3-18 16 ビット演算命令のマシン・サイクル数とステート数

命 令	REG	REGISTER					
		BC	DE	HL	SP	IX	IY
ADD	HL	M = 3 T = 11 (12)					
	IX	M = 4 T = 15 (17)					
	IY						M = 14 T = 15 (17)
ADC	HL	M = 4 T = 15 (17)					
SBC	HL						
INC		M = 1 T = 6 (7)				M = 2 T = 10 (12)	
DEC							

注) ・ M はマシン・サイクル

・ T はステート。

・ カッコ内の数は MSX の場合のステート数。

表3-19 16ビット演算命令によるフラグの変化

命 令	フラグ・レジスタ						
	S	Z	H	P/V	N	CY	
ADD HL, SS; ADD IX, SS; ADD IY, SS	.	.	x	x	x	.	0
ADC HL, SS	↑	↑	x	x	x	V	0
SBC HL, SS	↑	↑	x	x	x	V	1

注) . = 実行結果の影響を受けない。
 0 = リセットされる。 1 = セットされる。 x = 不定
 ↑ = 実行結果の影響を受ける。
 ・ 命令中のSSは16ビット・レジスタ(BC, DE, HL, SP, IX, IY)のこと。
 ・ P/Vフラグ
 オーバーフローあり → V = 1, オーバーフローなし → V = 0

① ADD 命令

ADD 命令はペア・レジスタ HL/インデックス・レジスタ IX, IY の内容に16ビット・レジスタの内容を加算し、結果をペア・レジスタ HL/インデックス・レジスタ IX, IY に格納します。たとえば、HL = 39C8H, BC = 43DFH のとき「ADD HL, BC」を実行すると HL = 7DA7H になります。

```

ペア・レジスタ HL = 39C8H      0011100111001000
ペア・レジスタ BC = 43DFH  +  0100001111011111
加算結果                    0111110110100111
ペア・レジスタ HL = 7DA7H
  
```

ADD 命令を実行すると、フラグは次のようになります。

CY: 結果が16ビットを超えたときセットされます (ビット15からのキャリーが入る)。

N: リセットされます。

H: 不定。

S, Z, P/V: どれも変化しません。

② ADC 命令

ADC 命令はキャリーフラグの内容も含めて16ビット加算をする命令です。たとえば、HL = 39C8H, BC = 43DFH のとき「ADC HL, BC」を実行すると、実行前のキャリーフラグが0なら HL = 7DA7H になり、キャリーフラグが1なら HL = 7DA8H になります。

ADC 命令実行後のフラグの状態は、

CY: 結果が16ビットを超えたときセットされます (ビット15からのキャリーが入る)。

N: リセットされます。

P/V: 結果が2の補数の範囲(10進数で-32768 ~ +32767)を超えたときセットされます。

H: 不定。

Z: ペア・レジスタ HL の全ビットがゼロになればセットされます。

S: 結果が2の補数の値として負(8000H ~ 0FFFFH) になったときセットされます。

たとえば、CY = 1, HL = 0A8C0H, DE = 9C7FH のとき「ADC HL, DE」を実行すると次のようになります。

```

ペア・レジスタ HL = 0A8C0H      1010100011000000
ペア・レジスタ DE = 9C7FH      + 1001110001111111
キャリー フラグ = 1          + 1
結果ペア・レジスタ HL = 4540H    1010001010100000
                                N 0  Z 0  CY 1  H 7  Z 0
                                S 0
  
```

③ SBC 命令

SBC 命令はキャリーフラグの内容も含めて16ビット減算をする命令です。

たとえば HL = 6C38H, DE = 12F9H のとき「SBC HL, DE」を実行すると、実行前のキャリーフラグが0なら HL = 593FH になり、キャリーフラグが1なら HL = 593EH になります。

SBC 命令実行後のフラグの状態は、

CY: 符号なし整数として見たとき、引かれる数(ペア・レジスタ HL の値)より引く数(16ビット・レジスタの値 + CY フラグ)のほうが大きければセットされます (ビット15からの borrow が入る)。

N: セットされます。

P/V: 結果が2の補数の範囲(10進数で-32768 ~ +32767)を超えた場合セットされます。

H: 不定。

Z: ペア・レジスタHLの全ビットがゼロになればセットされます。

S: 結果が2の補数の値として負(8000H ~ 0FFFFH) になったときセットされます。

たとえば, CY=0, HL=76F3H, BC=2CC8Hのとき「SBC HL, DE」を実行すると次のようになります。

ペア・レジスタHL = 76F3H 0111011011110011
 ペア・レジスタDE = 2CC8H 0010110011001000
 キャリー フラグ = 0 - 0
 結果ペア・レジスタHL = 4A2BH 0100101000101011
 N[?] Z[0] CY[0] S[0] H[?] Z[0]

④ INC 命令

ペア・レジスタ(BC, DE, HL), スタック・ポインタ(SP), インデックス・レジスタ(IX, IY)の内容に1を加える命令です。

この命令を実行してもフラグには影響を与えません。

⑤ DEC 命令

ペア・レジスタ(BC, DE, HL), スタック・ポインタ(SP), インデックス・レジスタ(IX, IY)の内容から1を引く命令です。

この命令を実行してもフラグには影響を与えません。

5. ローテイト, シフト

ローテイトに関する命令には,

RLCA (Rotate Left Circular Accumulator)
 RRCA (Rotate Right Circular Accumulator)
 RLA (Rotate Left Accumulator)
 RRA (Rotate Right Accumulator)
 RLC (Rotate Left Circular)

RRC (Rotate Right Circular)

RL (Rotate Left)

RR (Rotate Right)

RLD (Rotate Digit Left)

RRD (Rotate Digit Right)

の10命令があります。

シフトに関する命令は,

SLA (Shift Left Arithmetic)

SRA (Shift Right Arithmetic)

SRL (Shift Right Logical)

の3命令です。

ローテイト, シフト命令のマシン・コードを表3-20に, マシン・サイクル数, ステート数を表3-21に, 命令実行後のフラグの変化を表3-22に示します。

表3-20 ローテイト, シフト命令

オペランド 命令	REGISTER								REG. INDIR.	INDEXED	
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	
"RLCA"	07										
"RRCA"	0F										
"RLA"	17										
"RRA"	1F										
"RLC opr"	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06	
"RRC opr"	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E	
"RL opr"	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16	
"RR opr"	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E	
"SLA opr"	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26	
"SRA opr"	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E	
"SRL opr"	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E	
"RLD"								ED 6F			
"RRD"								ED 67			

注)・RLCA, RRCA, RLA, RRA, RLD, RRD はオペランドがない。

・dはディスプレイメント (2の補数-128~+127)

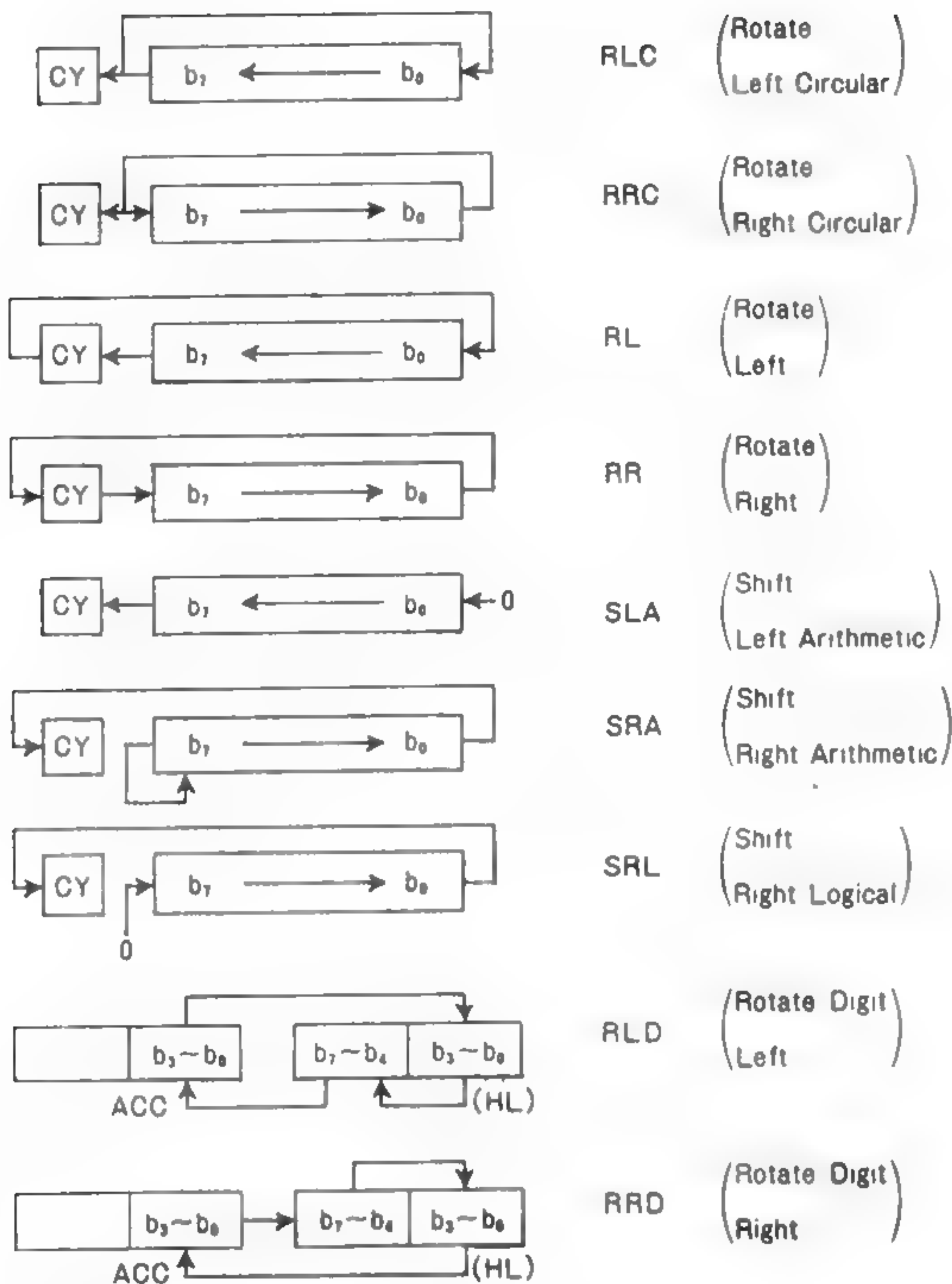


表3-21

ローテイト、シフト命令のマシン・サイクル数とステート数

オペランド 命令	REGISTER							REG. INDIR.	INDEXED	
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
RLCA	M = 1 T = 4 (5)									
RRCA										
RLA										
RRA										
RLC opr	マシン・サイクル M = 2 ステート T = 8 (10)							M = 4 T = 15 (17)	M = 6 T = 23 (25)	
RRC opr										
RL opr										
RR opr										
SLA opr										
SRA opr										
SRL opr								M = 5 T = 13 (20)		
RLD										
RRD										

注)・Mはマシン・サイクル

・Tはステートのこと。

・カッコ内の数はMSXの場合のステート数。

表3-22 ローテイト、シフト命令によるフラグの変化

命令		フラグ・レジスタ							
		S	Z	H	P/V	N	CY		
RLCA; RRCA; RLA; RRA		・	・	×	0	×	・	0	↑
RLC s; RRC s; RL s; RR s		↑	↑	×	0	×	P	0	↑
SLA s; SRA s; SRL s		操作されたレジスタ、メモリの状態を示します。							
RLD; RRD		↑	↑	×	0	×	P	0	・
		アキュムレータAの状態を示します。							

注)・=実行結果の影響を受けない。

0=リセットされる。×=不定

↑=実行結果の影響を受ける。

・命令中のsはオペランドのことです。

・P/Vフラグ

偶数パリティ→P=1, 奇数パリティ→P=0

ローテイト命令(RLD, RRD命令を除く)

ローテイト命令は、対象レジスタ/メモリを環状につなげて回転させる命令です。キャリーフラグを環に含める命令もあります。⁷⁴⁾

ローテイト命令にはアキュムレータAだけを対象にした命令と、8ビット・レジスタ/メモリの内容に対して操作できる命令とがあります。前者の命令と後者の命令は機能的には同じなのですが、フラグに対する影響のしかたが異なります。⁷⁵⁾

ローテイト・ディジット命令

ペア・レジスタHLが示すメモリの内容とアキュムレータAの下位4ビットを左右に4ビット回転させる命令です。

たとえばA=57H, HL=0D300H, メモリ0D300H番地の内容を82Hとして「RLD」命

① ローテイト命令に関する詳細は、2章3.3 (31ページ)を参照。
② たとえばA=98Hのとき、RLCAとRLC Aとでは、次のように異なる。

・RLCA命令

アキュムレータ		フラグ・レジスタ							
		S	Z	H	P/V	N	CY		
実行前	b7 b0 1 0 0 1 1 0 0 0	1	0	0	0	0	1	0	0
実行後	b7 b0 0 0 1 1 0 0 0 1	1	0	?	?	?	1	0	1

S, Z, P/Vフラグは不変

・RLC A命令

アキュムレータ		フラグ・レジスタ							
		S	Z	H	P/V	N	CY		
実行前	b7 b0 1 0 0 1 1 0 0 0	1	0	0	0	0	1	0	0
実行後	b7 b0 0 0 1 1 0 0 0 1	0	0	?	?	?	0	0	1

Sフラグにはビット7の内容。

ZフラグはA=0でないのでリセット。

P/Vフラグにはパリティ(奇数なので0)が入る。

令を実行すると、次のようになります。

	アキュムレータ	ペア・レジスタ HL	メモリ 0D300 番地								
	$b_7 \sim b_6$ $b_5 \sim b_0$	H L	$b_7 \sim b_6$ $b_5 \sim b_0$								
実行前	<table border="1"><tr><td>5</td><td>7</td></tr></table>	5	7	<table border="1"><tr><td>D</td><td>3</td><td>0</td><td>0</td></tr></table>	D	3	0	0	<table border="1"><tr><td>8</td><td>2</td></tr></table>	8	2
5	7										
D	3	0	0								
8	2										
	$b_7 \sim b_6$ $b_5 \sim b_0$	H L	$b_7 \sim b_6$ $b_5 \sim b_0$								
実行後	<table border="1"><tr><td>5</td><td>8</td></tr></table>	5	8	<table border="1"><tr><td>D</td><td>3</td><td>0</td><td>0</td></tr></table>	D	3	0	0	<table border="1"><tr><td>2</td><td>7</td></tr></table>	2	7
5	8										
D	3	0	0								
2	7										

この命令はメモリ上に連続して記憶されているBCD表現の10進数を、10進数の桁単位でシフトするときに使われます。

例) 0D301H, 0D300Hに記憶されている4桁のBCDを右へ1桁シフトする。

プログラム	アキュムレータ	メモリ 0D301H 番地	メモリ 0D300 番地
XOR A	?	1	2
LD HL, 0D301H	?	3	4
RRD			
DEC HL			
RRD			
	0 4	0 1	2 3

3 シフト命令

シフト命令は8ビット・レジスタ/メモリの内容に対してシフトを行なう命令です。⁽⁷⁴⁾

6. ビット操作, フラグ操作

ビット操作を行なう命令には、

BIT (test BIT)

RES (RESet bit)

SET (SET bit)

の3命令があります。

フラグ操作を行なう命令は、

CCF (Complement Carry Flag)

SCF (Set Carry Flag)

の2命令です。

1 ビット操作命令

表3-23がビット操作命令のマシン・コードです。ビット操作命令はレジスタ/任意のメモリの指定ビットをセット/リセットまたはテストすることができる命令です。⁽⁷⁷⁾ テストの場合、そのビットが0か1かの状態はZフラグに設定されます。マシン・サイクル数、ステート数を表3-24に、表3-25にはBIT命令実行後のフラグの状態を示します。SET, RES命令ではフラグは変化しません。下図は命令中で指定するビット番号と位置を示しています。

MSB	7	6	5	4	3	2	1	0	LSB
-----	---	---	---	---	---	---	---	---	-----

表3-24

ビット操作命令のマシン・サイクル数とステート数

	REGISTER	REG. INDIR.	INDEXED
	A B C D E H L	(HL)	(IX+d) (IY+d)
BIT	マシン・サイクル M=2 ステート T=8 (10)	M=3 T=12 (14)	M=5 T=20 (22)
RES		M=4 T=15 (17)	M=6 T=23 (25)
SET			

注) ・Mはマシン・サイクル
・Tはステートのこと。
・カッコ内の数はMSXの場合のステート数。

表3-25 ビット操作命令によるフラグの変化

命 令	S	Z	H	P/V	N	CY
BIT b, s	x	↑	x	1	x	0

↑
指定ビットが0ならセットされる。

注) ・=実行結果の影響を受けない。
0=リセットされる。 1=セットされる。 x=不定
↑=実行結果の影響を受ける。
・命令中のbはビット番号, Sはオペランドのこと。

⑦シフト命令に関する詳細は、2章3.3(31ページ)を参照。
⑧指定ビットをセットするとは、そのビットを「1」にすること。同様にリセットするとは「0」にすること。テストす

るとは、指定ビットの状態が「0」であるか「1」であるかを確かめること。

表3-23 ビット操作命令

	第1 オペランド	第 2 オペランド									
		R E G I S T E R							REG. INDIR.	INDEXED	
		B I T	A	B	C	D	E	H	L	(HL)	(IX+d)
TEST BIT "BIT opr 1, opr 2"	0	CB 4 7	CB 4 0	CB 4 1	CB 4 2	CB 4 3	CB 4 4	CB 4 5	CB 4 6	DD CB 4 ^d 6	FD CB 4 ^d 6
	1	CB 4 F	CB 4 8	CB 4 9	CB 4 A	CB 4 B	CB 4 C	CB 4 D	CB 4 E	DD CB 4 ^d E	FD CB 4 ^d E
	2	CB 5 7	CB 5 0	CB 5 1	CB 5 2	CB 5 3	CB 5 4	CB 5 5	CB 5 6	DD CB 5 ^d 6	FD CB 5 ^d 6
	3	CB 5 F	CB 5 8	CB 5 9	CB 5 A	CB 5 B	CB 5 C	CB 5 D	CB 5 E	DD CB 5 ^d E	FD CB 5 ^d E
	4	CB 6 7	CB 6 0	CB 6 1	CB 6 2	CB 6 3	CB 6 4	CB 6 5	CB 6 6	DD CB 6 ^d 6	FD CB 6 ^d 6
	5	CB 6 F	CB 6 8	CB 6 9	CB 6 A	CB 6 B	CB 6 C	CB 6 D	CB 6 E	DD CB 6 ^d E	FD CB 6 ^d E
	6	CB 7 7	CB 7 0	CB 7 1	CB 7 2	CB 7 3	CB 7 4	CB 7 5	CB 7 6	DD CB 7 ^d 6	FD CB 7 ^d 6
	7	CB 7 F	CB 7 8	CB 7 9	CB 7 A	CB 7 B	CB 7 C	CB 7 D	CB 7 E	DD CB 7 ^d E	FD CB 7 ^d E
RESET BIT "RES opr 1, opr 2"	0	CB 8 7	CB 8 0	CB 8 1	CB 8 2	CB 8 3	CB 8 4	CB 8 5	CB 8 6	DD CB 8 ^d 6	FD CB 8 ^d 6
	1	CB 8 F	CB 8 8	CB 8 9	CB 8 A	CB 8 B	CB 8 C	CB 8 D	CB 8 E	DD CB 8 ^d E	FD CB 8 ^d E
	2	CB 9 7	CB 9 0	CB 9 1	CB 9 2	CB 9 3	CB 9 4	CB 9 5	CB 9 6	DD CB 9 ^d 6	FD CB 9 ^d 6
	3	CB 9 F	CB 9 8	CB 9 9	CB 9 A	CB 9 B	CB 9 C	CB 9 D	CB 9 E	DD CB 9 ^d E	FD CB 9 ^d E
	4	CB A 7	CB A 0	CB A 1	CB A 2	CB A 3	CB A 4	CB A 5	CB A 6	DD CB A ^d 6	FD CB A ^d 6
	5	CB A F	CB A 8	CB A 9	CB A A	CB A B	CB A C	CB A D	CB A E	DD CB A ^d E	FD CB A ^d E
	6	CB B 7	CB B 0	CB B 1	CB B 2	CB B 3	CB B 4	CB B 5	CB B 6	DD CB B ^d 6	FD CB B ^d 6
	7	CB B F	CB B 8	CB B 9	CB B A	CB B B	CB B C	CB B D	CB B E	DD CB B ^d E	FD CB B ^d E
SET BIT "SET opr 1, opr 2"	0	CB C 7	CB C 0	CB C 1	CB C 2	CB C 3	CB C 4	CB C 5	CB C 6	DD CB C ^d 6	FD CB C ^d 6
	1	CB C F	CB C 8	CB C 9	CB C A	CB C B	CB C C	CB C D	CB C E	DD CB C ^d E	FD CB C ^d E
	2	CB D 7	CB D 0	CB D 1	CB D 2	CB D 3	CB D 4	CB D 5	CB D 6	DD CB D ^d 6	FD CB D ^d 6
	3	CB D F	CB D 8	CB D 9	CB D A	CB D B	CB D C	CB D D	CB D E	DD CB D ^d E	FD CB D ^d E
	4	CB E 7	CB E 0	CB E 1	CB E 2	CB E 3	CB E 4	CB E 5	CB E 6	DD CB E ^d 6	FD CB E ^d 6
	5	CB E F	CB E 8	CB E 9	CB E A	CB E B	CB E C	CB E D	CB E E	DD CB E ^d E	FD CB E ^d E
	6	CB F 7	CB F 0	CB F 1	CB F 2	CB F 3	CB F 4	CB F 5	CB F 6	DD CB F ^d 6	FD CB F ^d 6
	7	CB F F	CB F 8	CB F 9	CB F A	CB F B	CB F C	CB F D	CB F E	DD CB F ^d E	FD CB F ^d E

注)・dはディスプレイメント(2の補数 -128~+127)

2 フラグ操作命令

フラグ・レジスタの内容を操作する命令で、キャリーフラグの反転、セットができます。表3-26にマシン・コード、マシン・サイクル数、ステート数を、表3-27に命令実行によるフラグの変化を示します。

表3-26
フラグ操作命令のマシン・コード、マシン・サイクル、ステート数

	マシン・コード	マシン・サイクル	ステート
Complement Carry Flag *CCF*	3 F	1	4 (5)
Set Carry Flag *SCF*	3 7		

注) ステートのカッコ内の数はMSXの場合のステート数。

表3-27
フラグ操作命令のフラグの変化

命 令	S	Z	H	P/V	N	CY
C C F	・	・	X	X	・	↓
S C F	・	・	X	0	・	↑

注) ・=実行結果の影響を受けない。
0 =リセットされる。
↑ =セットされる。
X =不定
↓ =実行結果の影響を受ける。

7. ジャンプ、コール、リターン命令

ここで扱う命令はプログラムの流れを変えるためのもの、つまりプログラム・カウンタ(PC)を操作するための命令です。普通の状態では命令を1つ実行するとその命令の長さだけPCを自動的に増加させ次の命令へ移りますが、ジャンプ、コール、リターン命令ではPCに「ある値」を入れることによって「ある値」の番地から実行させることができます。

ジャンプ、コール、リターン命令は実行してもフラグの内容を変化させません。

1 ジャンプ命令

ジャンプ命令にはジャンプ先のアドレスを指定する方法(アドレッシング)⁷⁸としてイミディエイト・エクステンド、レジスタ・インダイレクト、レラティブの3種類があります。また、ジャンプ命令には、フラグの状態を調べて条件が合ったときだけジャンプし、条件が合わないときはジャンプせずに次の命令へ進む条件ジャンプと、フラグの状態には関係なくジャンプする無条件ジャンプとがあります。

表3-28にてマシン・サイクル数とステート数を、表3-30にはジャンプ命令のマシン・コードを示します。

表3-28
ジャンプ、コール、リターン命令のマシン・サイクルとステート数

	条 件 成 立		条件不成立	
	マシン・サイクル	ステート	マシン・サイクル	ステート
J P nn	3	10 (11)		
J P cond, nn	3	10 (11)	3	10 (11)
J R nn	3	12 (13)		
J R cond, nn	3	12 (13)	2	7 (8)
J P (HL)	1	4 (5)		
J P (IX)	2	8 (10)		
J P (IY)				
D J N Z nn	3	13 (14)	2	8 (9)
C A L L nn	5	17 (18)		
C A L L cond, nn	5	17 (18)	3	10 (11)
R E T	3	10 (11)		
R E T cond	3	11 (12)	1	5 (6)
R E T I	4	14 (16)		
R E T N				

注) ・条件成立とはジャンプ、コール、リターンの動作を行なうこと。
・条件不成立とは指定アドレスへのジャンプ、コール、リターンを行なわないこと。
・ステートのカッコ内の数はMSXの場合のステート数。

78 アドレッシングの種類と区別についての詳細は、3章2.1 (40ページ)を参照。

表3-30 ジャンプ命令

CONDITION			UN- COND	CARRY *C*	NON CARRY *NC*	ZERO *Z*	NON ZERO *NZ*	PARITY EVEN *PE*	PARITY ODD *PO*	SIGN NEG *M*	SIGN POS *P*	REG B ≠ 0
JUMP *JP nn*	IMMED. EXT.	nn	C _{nn} ³									
JUMP *JP cond, nn*				D A _{nn}	D 2 _{nn}	C A _{nn}	C 2 _{nn}	E A _{nn}	E 2 _{nn}	F A _{nn}	F 2 _{nn}	
JUMP *JR nn*	RELATIVE e ← nn - \$	PC + e	I 8 _{e-2}									
JUMP *JR cond, nn*				3 8 _{e-2}	3 0 _{e-2}	2 8 _{e-2}	2 0 _{e-2}					
JUMP *JP (HL)*	REG. INDIR.	(HL)	E 9									
JUMP *JP (IX)*		(IX)	D D E 9									
JUMP *JP (IY)*		(IY)	F D E 9									
DECREMENT B, JUMP IF NON ZERO *DJNZ nn*	RELATIVE e ← nn - \$	PC + e										I 0 _{e-2}

注)・表中の e はディスプレイシメント (2 の補数で -128 ~ +127) 例えば e は次のように求める。
 JR 0E300H をメモリの 0E2E0H に格納するとして
 nn = 0E300H, \$ = 0E2E2H となり, e = nn - \$ = 0E300H - 0E2E2H = 1EH となる。

① イミディエイト・エクステンド・アドレッシング (オペランドにジャンプ先のアドレスを直接書く)

このアドレッシングには無条件ジャンプと条件ジャンプの 2 つの命令があり,
 無条件ジャンプ = JP ジャンプ先アドレス
 条件ジャンプ = JP 条件名, 条件成立時のジャンプ先アドレス
 と表わします。

条件には次のようなものがあります。

条件名	成立条件
C (Carry)	CY フラグ = 1
NC (Non Carry)	CY フラグ = 0
Z (Zero)	Z フラグ = 1
NZ (Non Zero)	Z フラグ = 0
PE (Parity Even)	P/V フラグ = 1 (パリティ偶数, またはオーバーフローあり)
PO (Parity Odd)	P/V フラグ = 0 (パリティ奇数, またはオーバーフローなし)
M (Sign Negative)	S フラグ = 1 (2 の補数で負)
P (Sign Positive)	S フラグ = 0 (2 の補数で正, またはゼロ)

② レジスタ・インダイレクト・アドレッシング (レジスタの内容をジャンプ先のアドレスとする)

このアドレッシングの命令は次の 3 命令です。

JP (HL)
 JP (IX)
 JP (IY)

この命令を実行すると 16 ビット・レジスタ (HL, IX, IY) の内容が PC に転送され, それぞれのレジスタが示すアドレスへ無条件にジャンプします。

③ レラティブ・アドレッシング (相対的にジャンプ先のアドレスを求める = 相対ジャンプ)

このアドレッシングの命令には次のようなものがあります。

無条件ジャンプ = JR ジャンプ先アドレス
 条件ジャンプ = JR 条件名, ジャンプ先アドレス
 ループ制御 = DJNZ ジャンプ先アドレス

このアドレッシングで使える条件名は, C,

NC, Z, NZの4つです。

相対ジャンプ命令のマシン語は2バイトで、1バイト目にオペコード、2バイト目にディスプレイメント(2の補数)が入ります。このディスプレイメントの値と1バイト目のオペコードのアドレスによってジャンプ先のアドレスを求めます。この命令でジャンプできる範囲は1バイト目のオペコードのアドレスをゼロとして+129~-126バイトの間です(図3-18)。

相対ジャンプの仲間にはDJNZ命令という変わった命令があります。この命令を実行するとCPUは初めにレジスタBをデクリメントし、 $B \neq 0$ ならジャンプし、 $B = 0$ ならジャンプせずにつぎの命令に進みます。これは、次の2つの命令群と同等の動作ですが、このDJNZ命令は実行後フラグを変化させません。

この命令は、プログラムの一部を一定回数ループするようなときに使います。ただし、ループ回数は最低で1回、最大で256回($B = 0$ として始めたとき)です。

DJNZジャンプ先アドレス = $\begin{cases} \text{DEC B} \\ \text{JR NZ, ジャンプ先アドレス} \end{cases}$

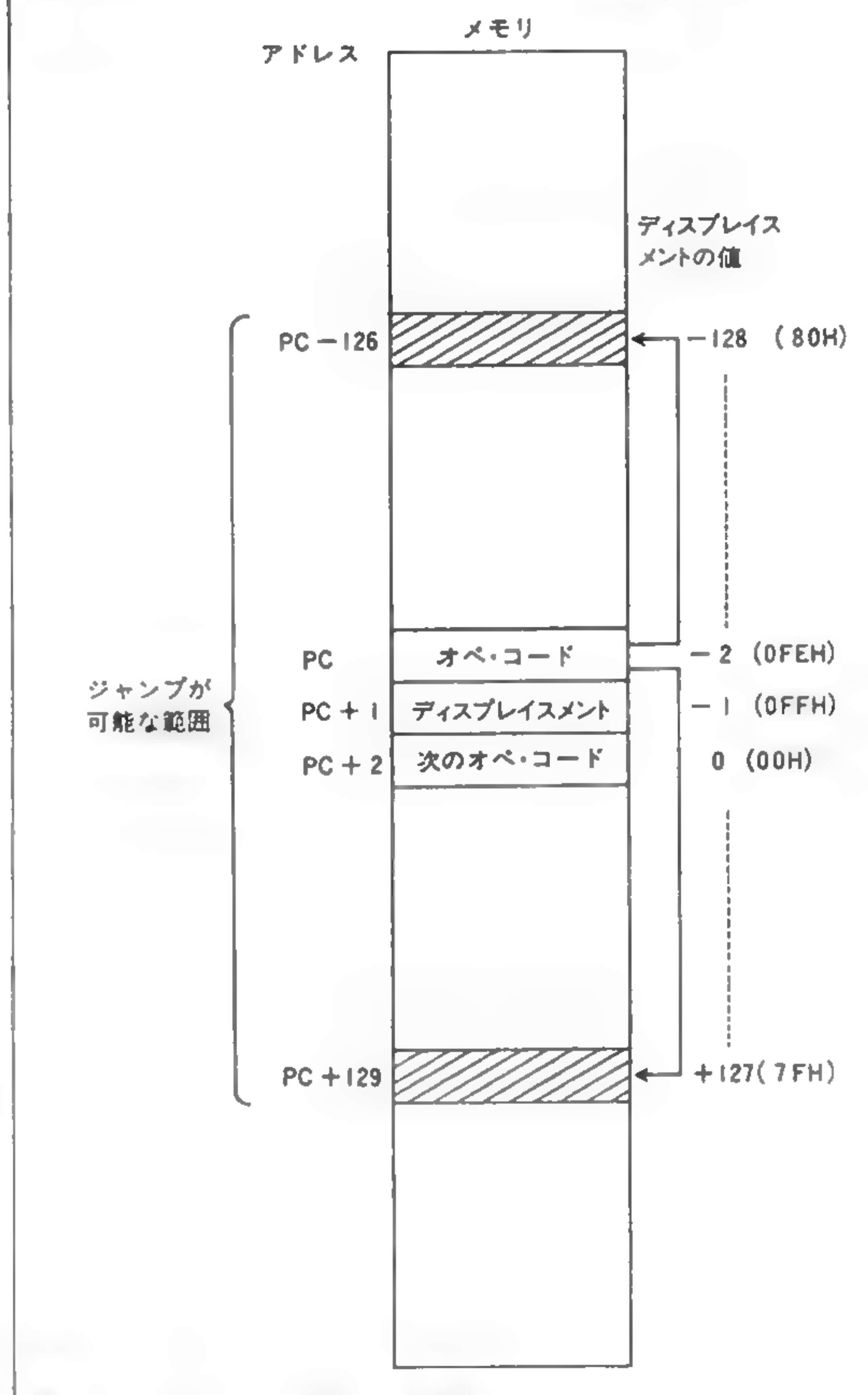
命令のバイト数	2 バイト	3 バイト
フラグの変化	なし	あり DEC Bのときフラグが変化する。
命令のステート数	$\begin{cases} \text{ジャンプあり: 13(14)} \\ \text{ジャンプなし: 8(9)} \end{cases}$	$\begin{cases} \text{ジャンプあり: 16(18)} \\ \text{ジャンプなし: 11(13)} \end{cases}$

注) カッコ内はMSXでのステート数。

⑩相対ジャンプ命令の次の命令が書かれているアドレス(すなわち相対ジャンプ命令が書かれているアドレス+2)にこのディスプレイメントを加算して、目的のジャンプ先アドレスを求める。

⑪同一プログラム中に同じ動作をさせる部分がいくつかある場合、それを1カ所に集めて必要時に呼び出せるようにしたものを「サブルーチン」と呼ぶ。プログラムの各部分をサブルーチン化することによってプログラム全体を短くすること

図3-18 レラティブ・ジャンプの範囲



● コール、リターン命令

コール命令は、サブルーチンへ実行を移すための命令で、リターン命令はサブルーチンから元のルーチンへ戻るための命令です。図3-19にサブルーチンを使ったときのプログラムの流れを示します。

表3-29はコール、リターン命令のマシン・コードです。表3-28 (P62参照) にマシン・サイクル数、ステート数を示します。

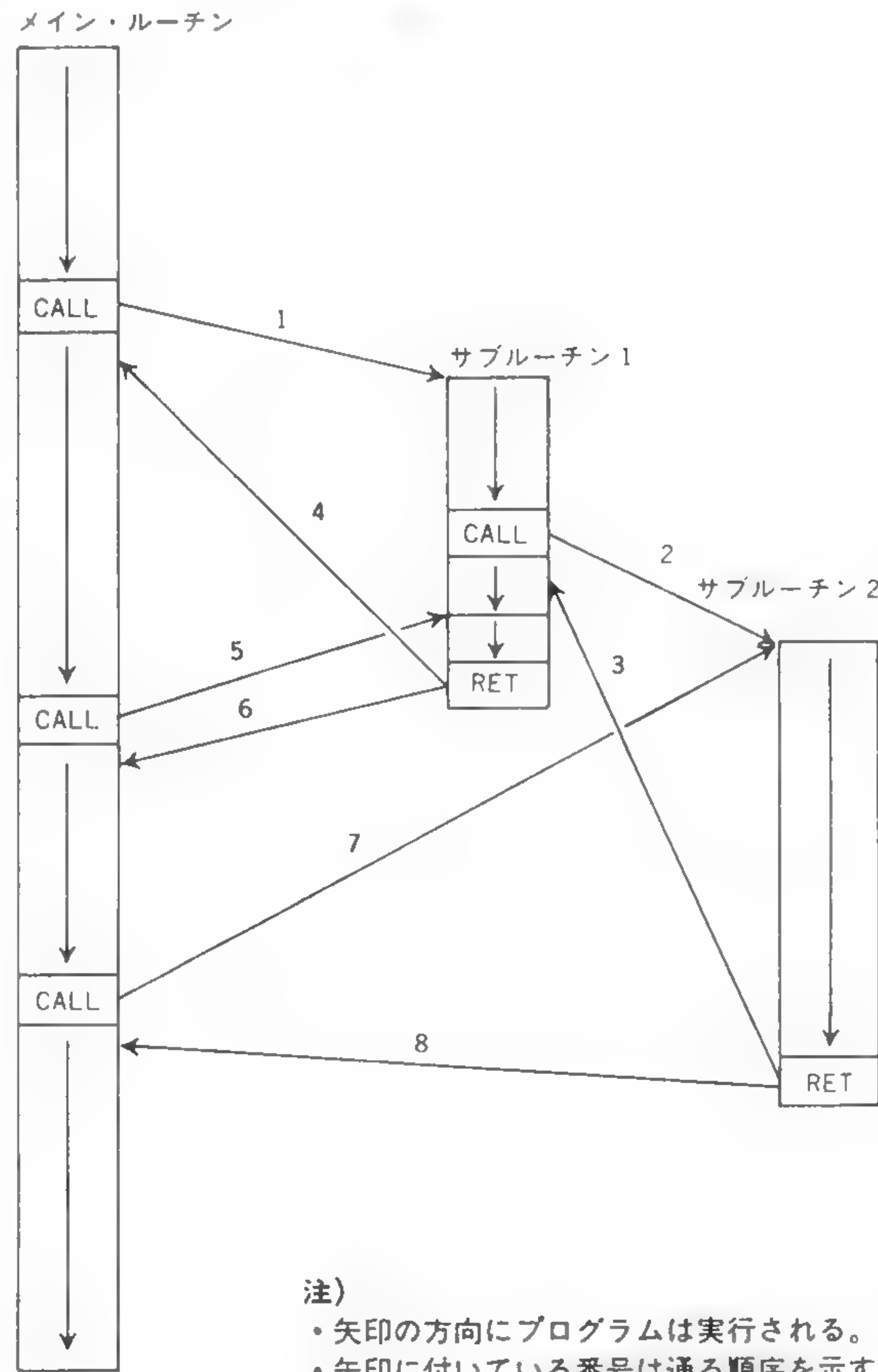
ができ、1カ所の修正がところどころに及ぶ心配もなくなる。これに対して、サブルーチンと呼んでいる元のプログラムを「メイン・ルーチン」という。

サブルーチンと呼び出す命令がコール命令、サブルーチンから戻る命令がリターン命令である。サブルーチンと呼び出したときのアドレスをスタックに置いておき、リターン命令でスタックから戻り番地を引っ張り出してきてジャンプするわけである。

表3-29 コール, リターン命令

命 令			CONDITION	UN-COND	CARRY "C"	NON CARRY "NC"	ZERO "Z"	NON ZERO "NZ"	PARITY EVEN "PE"	PARITY ODD "PO"	SIGN NEG "M"	SIGN POS "P"
CALL "CALL nn"	IMMED. EXT.	nn		C D n n								
CALL "CALL cond, nn"					D C n n	D 4 n n	C C n n	C 4 n n	E C n n	E 4 n n	F C n n	F 4 n n
RETURN "RET "	REG. INDIR.	(SP) (SP+1)		C 9								
RETURN "RET cond"					D 8	D 0	C 8	C 0	E 8	E 0	F 8	F 0
RETURN FROM INT "RETI"				E D 4 D								
RETURN FROM NON MASKABLE INT "RETN"				E D 4 5								

図 3-19 サブルーチンを使ったときのプログラムの流れ



①コール命令

コール命令にもジャンプ命令と同じように無条件コールと条件コールの2種類があり、
無条件コール= CALL コール先アドレス
条件コール= CALL 条件名, コール先アドレス
と表わします。

コール命令の動作は、初めにリターン・アドレスをスタックにプッシュし、次にコール先アドレスへジャンプします (図3-20)。

②リターン命令

リターン命令は、スタックからリターン・アドレスをポップしてPCに格納します。この動作によりコールしたルーチンへ戻ることができます (図3-21)。

リターン命令には次の3種類の命令があります。

i) RET 命令

RET 命令には無条件リターンと条件リターンがあり、

無条件リターン= RET

条件リターン = RET 条件名

と表わします。

ii) RETI 命令

割込みルーチンの最後に使います。この命令により、割込みで中断されたルーチンへ戻ります。この命令はI/OにZ80A専用LSIを使っているときには必要ですが、もし使っていなければこの命令の代わりにRET命令を使ってもかまいません。

iii) RETN命令

ノンマスカブル割込み(NMI)ルーチンの最後⁽⁸¹⁾に使います。この命令により、割込みで中断されたルーチンへ戻ります。ノンマスカブル割込みルーチンでは必ずRETN命令で割込みルーチンを終了させなければならず、RET命令⁽⁸²⁾やRETI命令で戻ってはいけません。

図 3-20 コール命令の動作

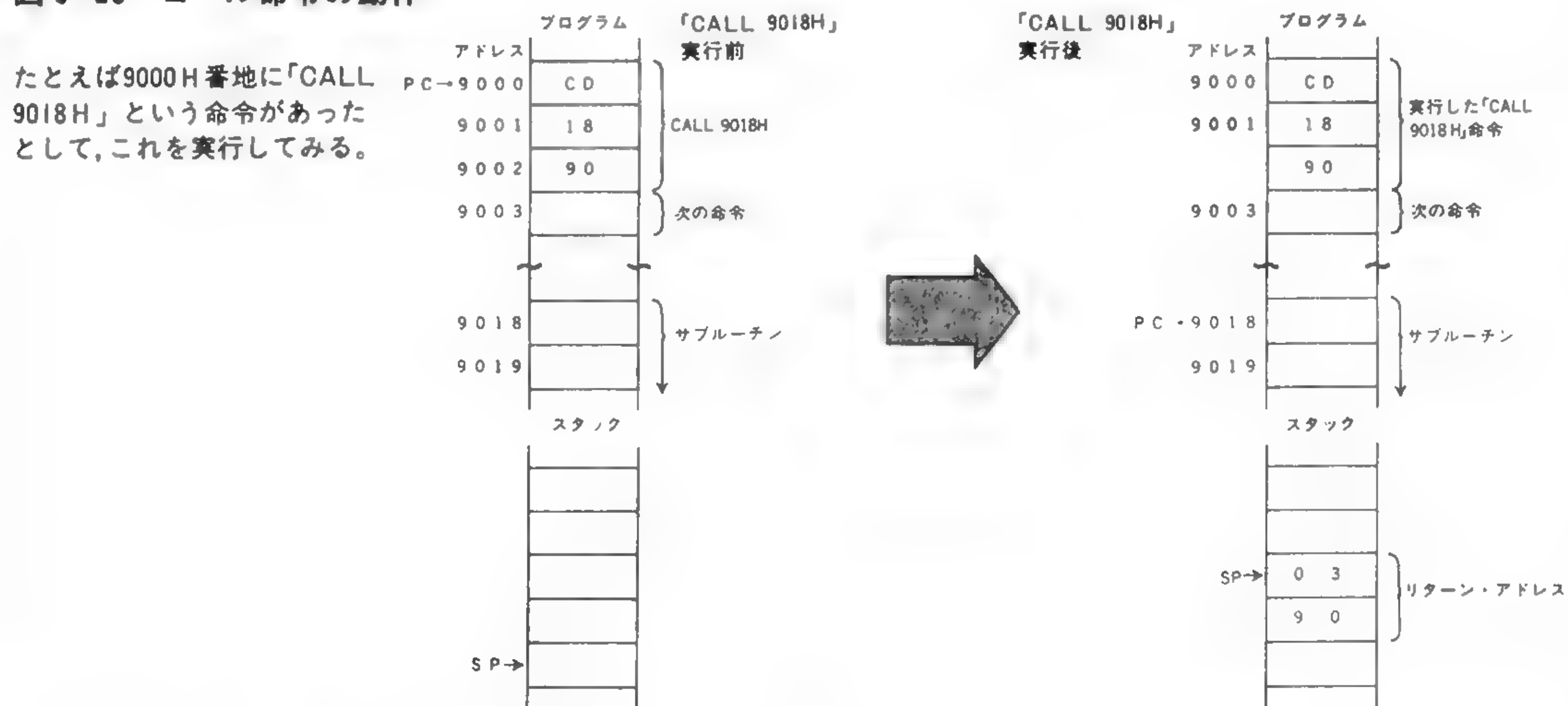
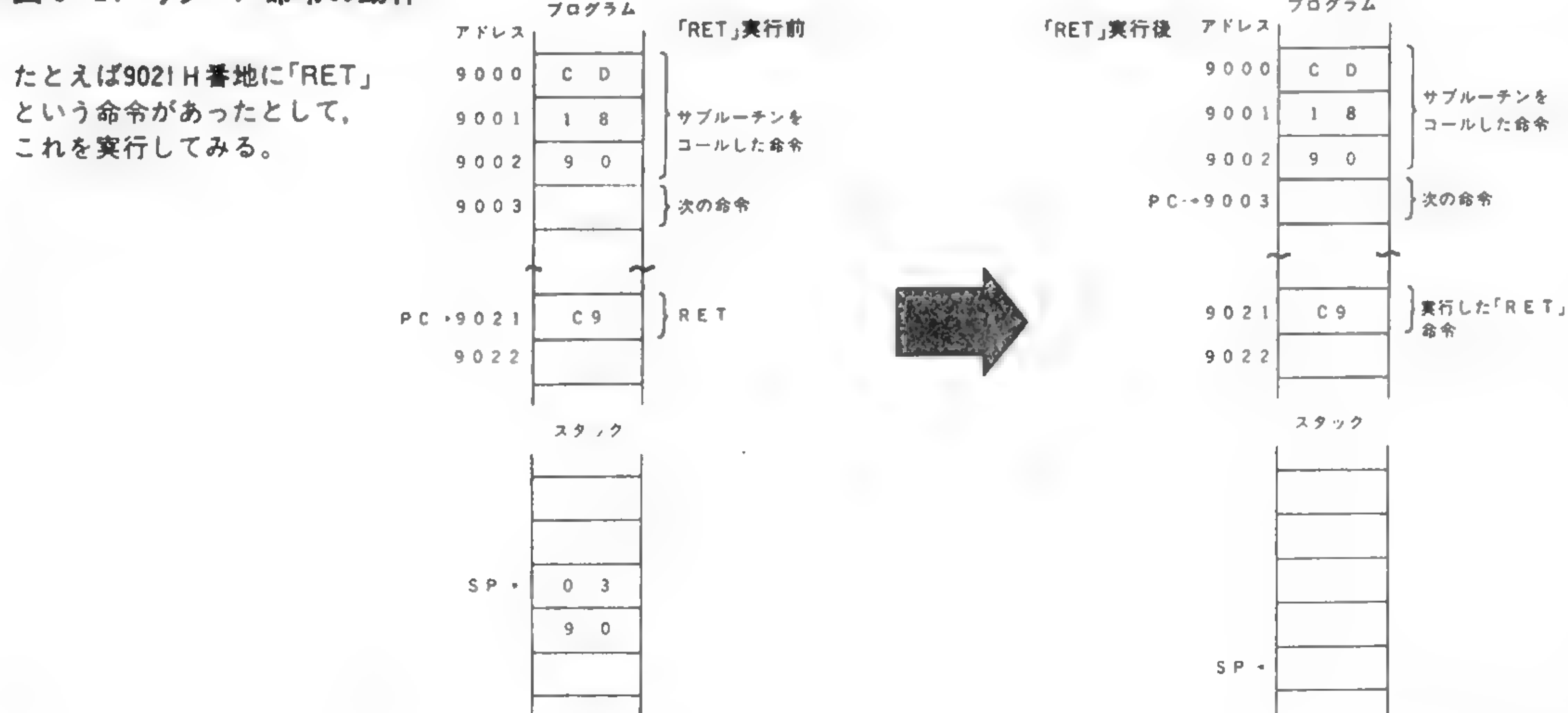


図 3-21 リターン命令の動作



④それは、RETN 命令が IFF (割込み許可フリップ・フロップ) と呼ばれるものをノンマスクابل割込みが発生する前の状態

に戻す動作をするからである。RET、RETI命令ではこの動作をしない。ただし、MSXではRETN命令が必要なノンマスクブル割込みを使っていないので、この命令は使わない。

3 リスタート命令

リスタート命令は、いわば1バイトのコール命令です。リスタート命令は表3-31に示すように8個あり、それぞれコールするアドレスが決まっています。

呼ぼうとするサブルーチンのアドレスがリスタート命令でコールできるアドレスなら、CALL命令を使うよりRST命令を使った方がメモリを2バイト節約できます。たとえば「CALL 18H」ならば「RST 18H」に変更することができます。

表3-31

リスタート命令のマシン・コード、マシン・サイクル、ステート数

			マシン・コード	マシン・サイクル	ステート
RESTART "RST addr"	CALL ADDRESS	00H	C7	3	11 (12)
		08H	CF		
		10H	D7		
		18H	DF		
		20H	E7		
		28H	EF		
		30H	F7		
		38H	FF		

注) ステートのカッコ内の数はMSXの場合のステート数。

8. 入出力命令

入出力命令は表3-32に示すIN, OUT命令と表3-35に示すブロック入出力命令の2種類があります。IN, OUT命令は、I/Oポートと8ビット・レジスタ(A, B~L)との間で1バイト入出力する命令です。ブロック入出力命令はメモリ上の連続したデータを1つのポートへ続けて出力したり、1つのポートからあるまとまったデータを続けて入力するための命令です。表3-33は入出力命令のマシン・サイクルとステート数、表3-34は入出力命令実行後のフ

ラグの状態を示します。また、表3-36はブロック入出力命令実行後のラグの状態を示します。

表3-32 入出力命令

		REGISTER						
		A	B	C	D	E	H	L
INPUT	"IN reg, (n)"	D B n						
	"IN reg, (c)"	ED 7 8	ED 4 0	ED 4 8	ED 5 0	ED 5 8	ED 6 0	ED 6 8
OUTPUT	"OUT (n), reg"	D 3 n						
	"OUT (c), reg"	ED 7 9	ED 4 1	ED 4 9	ED 5 1	ED 5 9	ED 6 1	ED 6 9

表3-33

入出力命令のマシン・サイクル数とステート数

		REGISTER						
		A	B	C	D	E	H	L
INPUT	INreg, (n)	M = 3 T = 11 (12)						
	INreg, (c)	マシン・サイクル M = 3 ステート T = 12 (14)						
OUTPUT	OUT(n), reg	M = 3 T = 11 (12)						
	OUT(c), reg	マシン・サイクル M = 3 ステート T = 12 (14)						

注) ・Mはマシン・サイクル
・Tはステートのことです。
・カッコ内の数はMSXの場合のステート数。

表3-34 入出力命令によるフラグの変化

		フラグ・レジスタ						
命	令	S	Z	H	P/V	N	CY	
IN reg, (c)		↓	↓	X	0	X	P 0	

注) ・=実行結果の影響を受けない。
0=リセットされる。 X=不定。
↑=実行結果の影響を受ける。
・命令中の reg はレジスタのことです。
・P/Vフラグ
偶数パリティ→P=1, 奇数パリティ→P=0

表3-36 ブロック入出力命令によるフラグの変化

		フラグ・レジスタ					
命	令	S	Z	H	P/V	N	CY
INI ; INDR ; OTIR ; OTDR		X	1	X	X	X	1
INI ; INDD ; OUTI ; OUTD		X	1	X	X	X	1

もし BC-1=0 ならば Z=1, その他 Z=0

注) ・=実行結果の影響を受けない。
↑=セットされる。 X=不定。
↑=実行結果の影響を受ける。

表3-35 ブロック入出力命令（INIR, INDR, INI, IND, OTIR, OTDR, OUTI, OUTD）

		マシン コード	マシン・ サイクル	ステート	動 作
BLOCK INPUT	"INIR"	ED B 2	B≠0のとき 5	B≠0のとき 21(23)	Memory(HL)←Port(c), Inc HL, Dec B, Repeat until B=0
	"INDR"	ED B A	B=0のとき 4	B=0のとき 16(18)	Memory(HL)←Port(c), Dec HL, Dec B, Repeat until B=0
	"INI"	ED A 2	4	16 (18)	Memory(HL)←Port(c), Inc HL, Dec B
	"IND"	ED A A			Memory(HL)←Port(c), Dec HL, Dec B
BLOCK OUTPUT	"OTIR"	ED B 3	B≠0のとき 5	B≠0のとき 21(23)	Port(c)←Memory(HL), Inc HL, Dec B, Repeat until B=0
	"OTDR"	ED B B	B=0のとき 4	B=0のとき 16(18)	Port(c)←Memory(HL), Dec HL, Dec B, Repeat until B=0
	"OUTI"	ED A 3	4	16 (18)	Port(c)←Memory(HL), Inc HL, Dec B
	"OUTD"	ED A B			Port(c)←Memory(HL), Dec HL, Dec B

注) ステートのカッコ内の数は MSX の場合のステート数。

1 IN, OUT 命令

IN, OUT 命令には次の 4 つの命令があります。

●IN A, (n)

この命令は、n で指定された入力ポートからアキュムレータ A に 1 バイトのデータを入力します。この命令はフラグに影響を与えません。

●IN reg, (C)

この命令は、レジスタ C（の内容）が示す入力ポートから 8 ビット・レジスタ（A, B～L）に 1 バイトのデータを入力します。この命令を実行するとフラグが次のように変化します。

- CY: 変化しません。
- N: リセットされます。
- P/V: 入力データの 8 ビットのうち、1 になっているビットの数が偶数ならセットされます。
- H: リセットされます。
- Z: 入力データが 00H ならセットされます。
- S: 入力データの MSB（ビット 7）が 1 ならセットされます。

●OUT (n), A

この命令は、アキュムレータ A の内容を、

n で指定された出力ポートへ 1 バイト出力します。この命令はフラグに影響を与えません。

●OUT (C), reg

この命令は、8 ビット・レジスタ（A, B～L）の内容をレジスタ C（の内容）が示す出力ポートへ 1 バイト出力します。この命令はフラグに影響を与えません。

2 ブロック入出力命令

ブロック入出力命令は次のレジスタを特定の目的に使って実行されます。

- B: バイト数カウンタ
 - C: I/O ポートのアドレス
 - HL: 入力の場合は入力データの転送先アドレス、出力の場合は出力データの格納アドレス
- これらのレジスタはブロック入出力命令実行前に設定しておきます。

1 INIR(InPut, Increment and Repeat)

INIR 命令は、レジスタ C が示す入力ポートからデータを入力し、ペア・レジスタ HL が示すメモリへ転送します。次にペア・レジスタ HL はインクリメントされ、レジスタ B はデクリメントされます。B = 0 になるまでこの動作を繰り返します。

2 INDR(INput, Decrement and Repeat)

INDR命令は、レジスタCが示す入力ポートからデータを入力し、ペア・レジスタHLが示すメモリへ転送します。次にペア・レジスタHLとレジスタBをデクリメントします。B=0になるまでこの動作を繰り返します。

3 INI(INput and Increment)

INI命令は、レジスタCが示す入力ポートからデータを入力し、ペア・レジスタHLが示すメモリへ転送します。次にペア・レジスタHLをインクリメントし、レジスタBをデクリメントします。なお、このときB=0になればZフラグがセットされます。

4 IND(INput and Decrement)

IND命令は、レジスタCが示す入力ポートからデータを入力し、ペア・レジスタHLが示すメモリへ転送します。次にペア・レジスタHLとレジスタBをデクリメントします。なお、このときB=0になればZフラグがセットされます。

5 OTIR(OuTput, Increment and Repeat)

OTIR命令は、ペア・レジスタHLが示すメモリの内容をレジスタCが示すポートへ出力します。次にペア・レジスタHLはインクリメントされ、レジスタBはデクリメントされます。B=0になるまでこの動作を繰り返します。

6 OTDR(OuTput, Decrement and Repeat)

OTDR命令は、ペア・レジスタHLが示すメモリの内容をレジスタCが示すポートへ出力します。次にペア・レジスタHLとレジスタBをデクリメントします。B=0になるまでこの動作を繰り返します。

7 OUTI(OUTput and Increment)

OUTI命令は、ペア・レジスタHLが示す

メモリの内容をレジスタCが示すポートへ出力します。次にペア・レジスタHLをインクリメントし、レジスタBをデクリメントします。なお、このときB=0になればZフラグがセットされます。

8 OUTD(OUTput and Decrement)

OUTD命令は、ペア・レジスタHLが示すメモリの内容をレジスタCが示すポートへ出力します。次にペア・レジスタHLとレジスタBをデクリメントします。なお、このときB=0になればZフラグがセットされます。

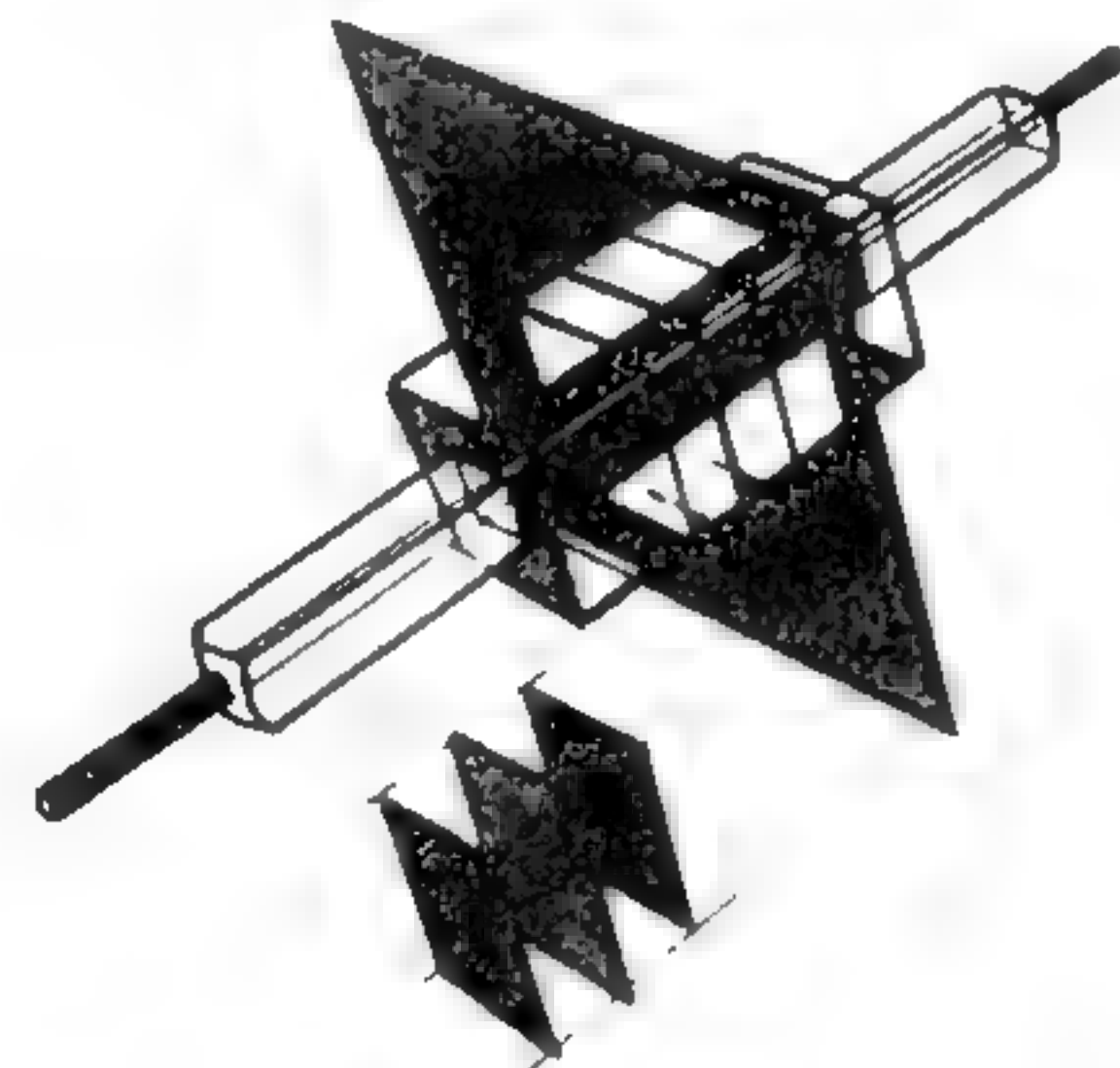
9. CPUコントロール

表3-37にCPUコントロール命令を示します。

表3-37 CPU コントロール命令

	マシン・コード	マシン・サイクル	ステート
NOP	0 0	1	4 (5)
HALT	7 6		
DISABLE INT *DI*	F 3		
ENABLE INT *EI*	F B		
SET INT MODE 0 *IM 0*	E D 4 6	2	8 (10)
SET INT MODE 1 *IM 1*	E D 5 6		
SET INT MODE 2 *IM 2*	E D 5 E		

注) ステートのカッコ内の数は MSX の場合のステート数。



1 NOP(No Operation)

NOP命令は名前の通り、実行しても何の動作もしません。ただ4ステート(MSXは5ステート、1.4 μ 秒)の時間を消費するだけです。

2 HALT(HALT)

HALT命令を実行すると、CPUはHALT命令が書かれているアドレスで停止します。この状態は、外部からのCPUリセット信号⁽⁸³⁾が⁽⁸³⁾入力されるか、外部からの割込み(INTまたはNMI)が入力されるまで続きます。

3 DI(Disable Interrupt)

マスカブル割込み(INT)による割込みを禁止します。これは、DI命令がIFF(割込み許可フリップ・フロップ)をリセットすることで行なわれます。IFFは、DI命令を実行しなくても割込み発生時に自動的にリセットされ、次の割込みが入るのを禁止しています。

4 EI(Enable Interrupt)

マスカブル割込み(INT)による割込みを可能にします。この命令によりIFFがセットされます。

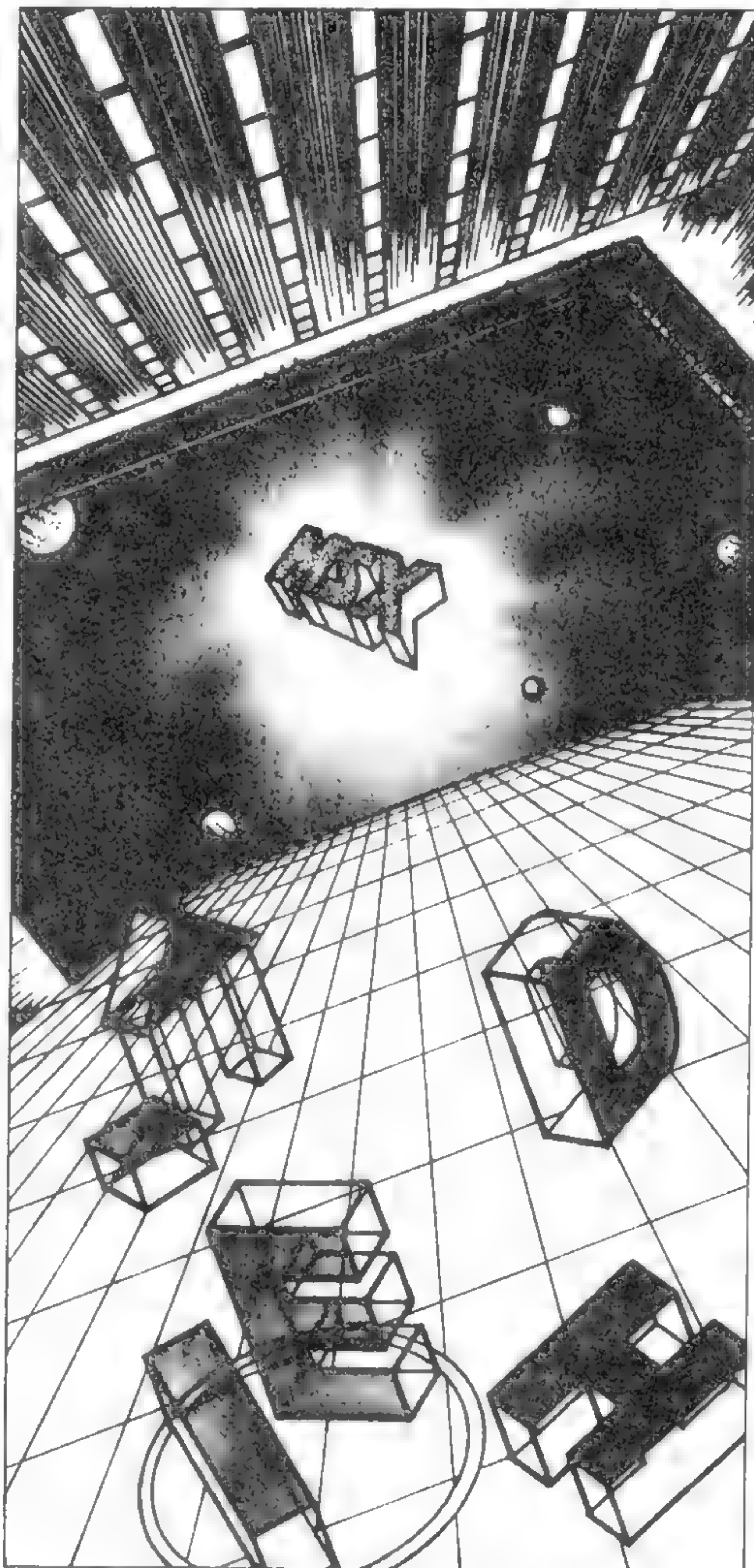
5 IM(Interrupt Mode)

割込みモードを設定する命令で、モードは0、1、2の3種類があります。

- ・モード0→8080A CPU⁽⁸⁴⁾と同じ割込み方法。
- ・モード1→割込み発生時に自動的にメモリの0038H番地⁽⁸⁵⁾をコールする。
- ・モード2→割込み発生時に割込み発生源から送られてくる8ビットのデータを

下位アドレスとし、レジスタIを上位アドレスとして求められたメモリ上の2バイトのデータをそのまま割込みルーチンの開始アドレスとしてコールする。

注意:MSXではモード1の割込みを使っています。使用者がこのIM命令で割込みモードを変更することは絶対にしないでください。



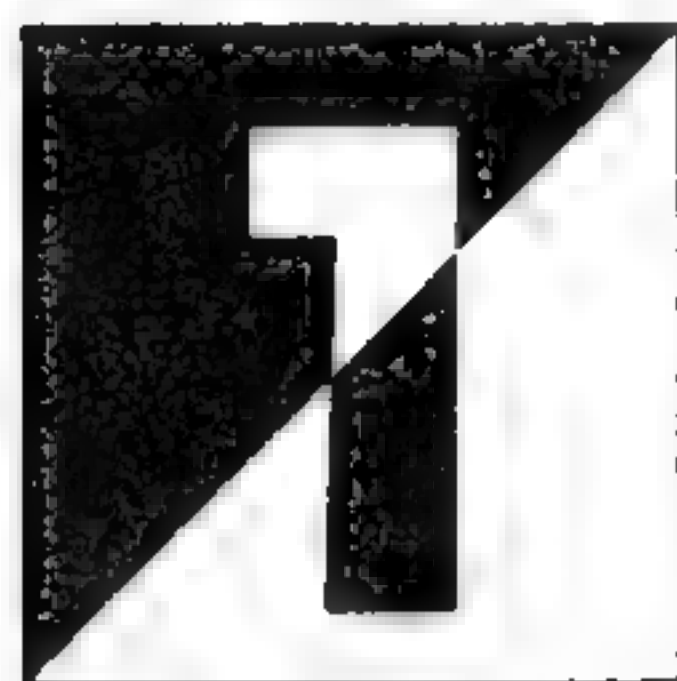
⑧ MSXにあるリセットボタンを押すと、この信号が出るようになっている(リセットボタンがないMSXもある)。

⑨ Z80の先祖に当たるCPU。

⑩ 「RST 38H」と同様な動作を行なう。

4章 モニタ・アセンブラ

この章では、MSX用に私が開発したマシン語モニタとセルフ・アセンブラの使用方法を述べるとともに、モニタ・アセンブラのプログラム・リストを掲載しました。このプログラム・リストをあなたのMSXに入力すれば、すぐにモニタやアセンブラによるマシン語プログラムの開発・デバッグができるようになります。



4 モニタ・アセンブラ

モニタ・アセンブラの概要

MSX用に作成したマシン語モニタには次のような機能があります。

- ①メモリ・ダンプ
- ②メモリの内容表示と変更
- ③レジスタの内容表示と変更
- ④マシン語プログラムの実行と指定アドレスでの停止
- ⑤アセンブラが出力したマシン語（これをオブジェクトという）のメモリへのロード

以上の機能を使って、作成したマシン語プログラムのデバッグ（プログラム中にある誤りを見つけ出し修正すること）を行ったり、16進数で表わされているマシン語プログラムやデータなどを直接メモリに入力することができます。

アセンブラは、MSX-BASICのエディタ⁸⁶⁾で作られたソース・プログラム（ニモニックで書かれたアセンブラ言語のプログラムのこと）をアセンブルし、VDPが使っていないVRAM⁸⁷⁾エリアにオブジェクトを出力します。アセンブルとはニモニックで書かれたアセンブラ言語のソース・プログラムをマシン語に翻訳する作業のことをいい、それをするプログラムのことをアセンブラといいます。

このMSX用のアセンブラは最大で8Kバイトのオブジェクトをつくることができます。ま

た、ソース・プログラムはRAM上に置かれているので、RAMのサイズが16Kバイトのシステムでは約300行、32Kバイトのシステムで約1300行のソース・プログラムを作成できます。ただし、この行数は1行16バイトとして求めた値ですので、実際の行数は多少上下します。

このモニタ・アセンブラにはストリング・サーチという機能があります。ストリング・サーチとは、ソース・プログラムの中から指定した文字列を含む行を探し出すものです。たとえばこのストリング・サーチを使って、ある特定の命令が使われている行を探し出すといったことができます。

モニタ・アセンブラのプログラムをカセットからロードするとき、モニタとアセンブラは1本のマシン語プログラムとしてロードされます。モニタだけを分離して使うこともできます。また、そうすることでアセンブラが格納されていたメモリ・エリアを自由に使えるようになります。図4-1はモニタ・アセンブラがロードされたときのメモリ・マップです。

図4-2はアセンブラが使うVRAMエリアのメモリ・マップです。アセンブラはスクリーンを40×24のテキスト・モードで使うように作られています⁸⁸⁾。ほかのスクリーン・モードではアセンブ

⑧BASICインタープリタには、BASICプログラムを翻訳・実行する狭義のインタープリタ部と、BASICプログラムの文を作成するエディタ部とが含まれている。プログラムを打ち込んだりリストを取ったりするときに使われるのがエディタ部、RUNさせるのがインタープリタ部である。インタープリタに限らず、アセンブラやコンパイラにもエディタが（内蔵させているとは限らないが）不可欠である。このMSX用アセンブラは独自のエディタを持たず、MSX-BASICのエディタを借

用してニモニックのソース・プログラムを作成する。こうすることで、アセンブラ自体のプログラム・サイズが小さくなり、セーブ、ロードもBASICと同様にできるので大変便利である。こうして作ったソース・プログラムは、当然ながらBASICのプログラムと同じ扱いができる。

⑨画面のデータやそれに類するデータを格納するためのRAM。最近のマシンでは、プログラムを格納するための本来のメインRAMとは別に何10Kバイトも持っているのが普通となって

ラは実行できません。また、デバッグの途中でスクリーンのモードを変更すると、VRAMに出力されたオブジェクトは壊されてしまいます。このモニタ・アセンブラを使うときは、スクリーンのモードに注意してください。

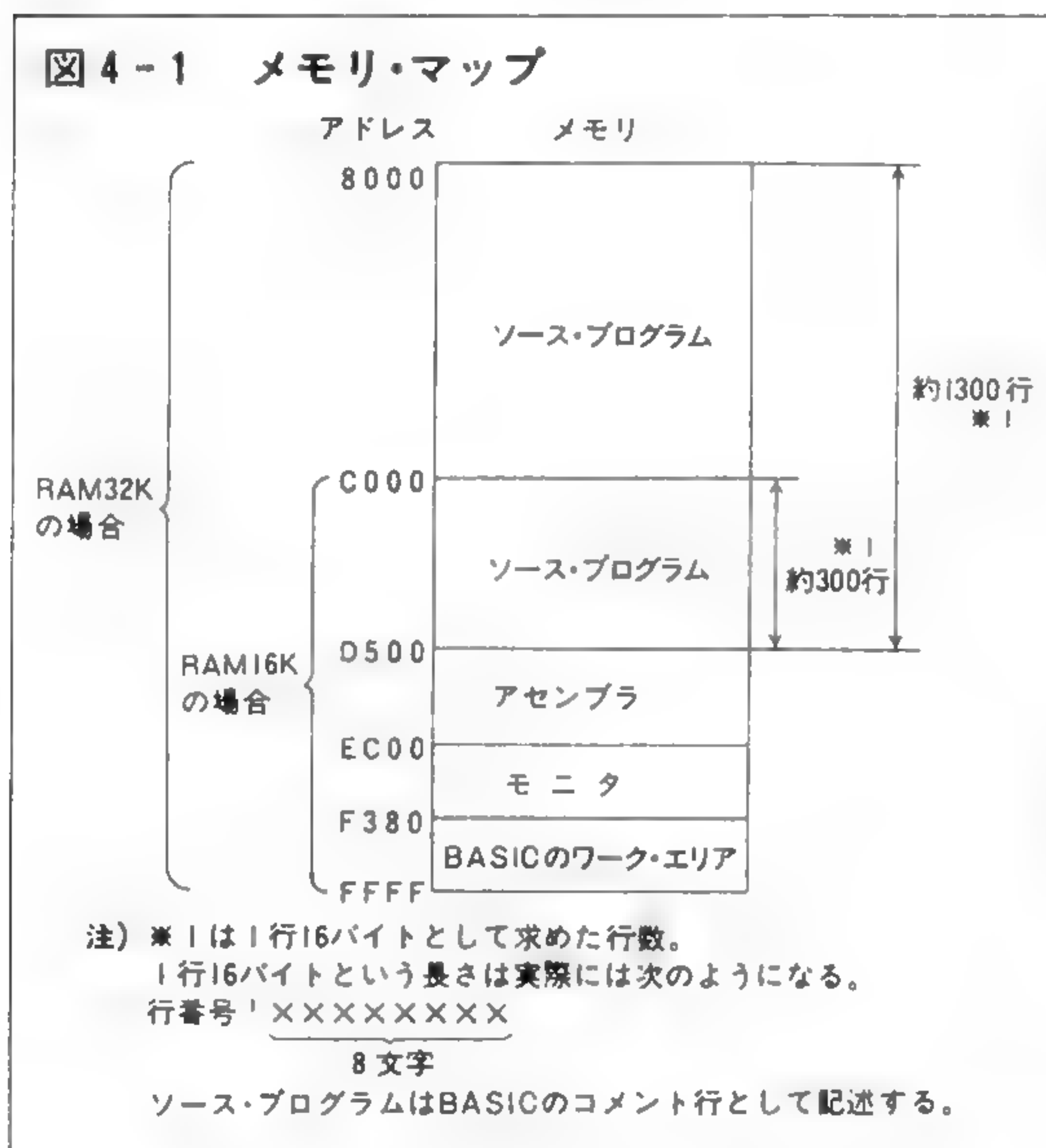
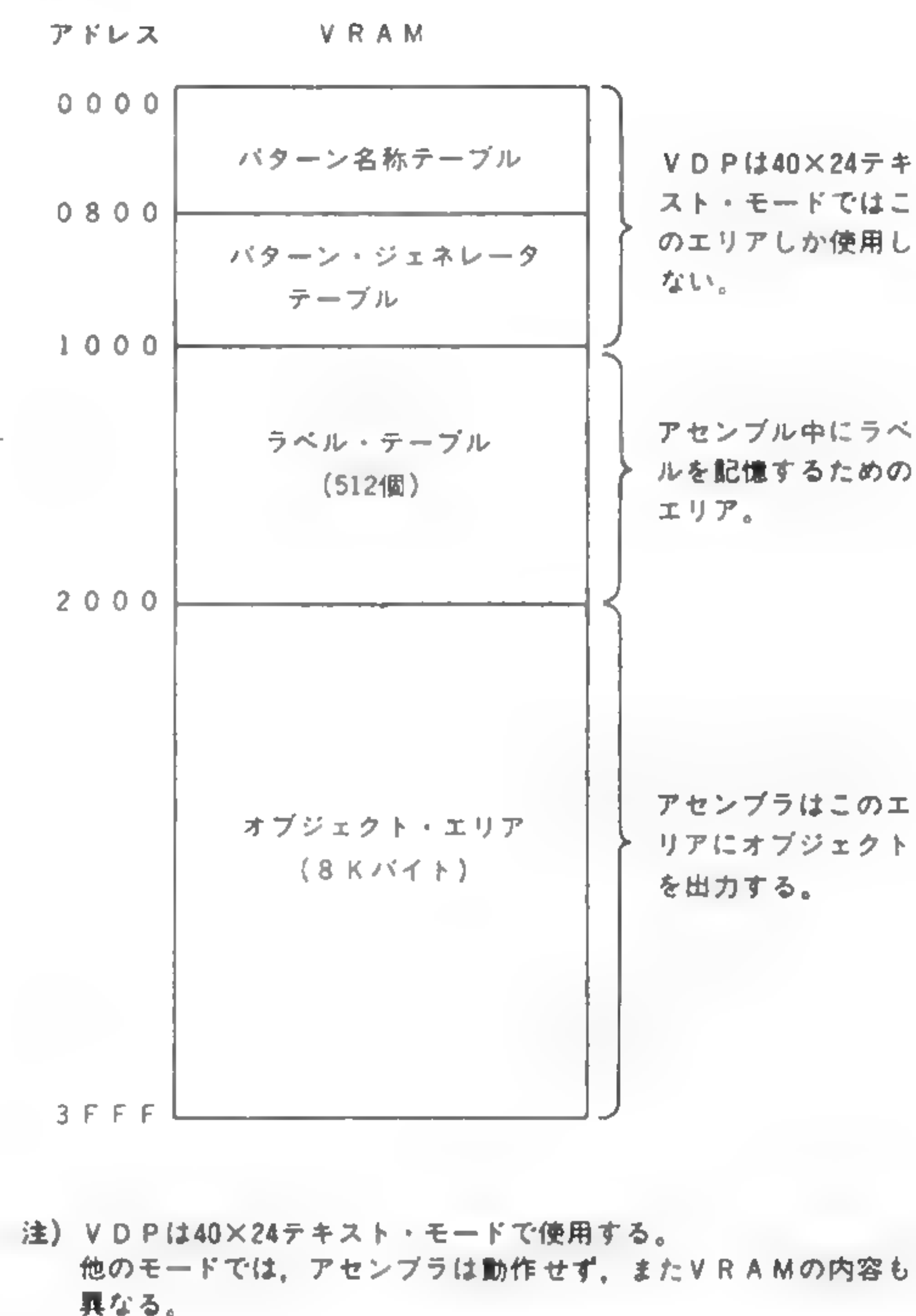


図4-2 VRAMのメモリ・マップ



4 モニタ・アセンブラ

起動方法

MSXの電源をONにして次の操作を行ない、モニタ・アセンブラをカセットからロードし、イニシャライズ（初期設定^⑧）します。

右記の操作により、モニタ・アセンブラのロードが終了すればモニタ・アセンブラはいつでも使える状態になります。

```
SCREEN 0 RETURN
CLEAR 0, &HD500 RETURN
MAXFILES=0 RETURN
BLOAD "CAS:", R RETURN
```

いる。Z80Aは8ビットマシンなので、CPUが直接管理できるメモリ空間は64Kバイトだが、バンク切り換えなどのテクニックを駆使してメインRAM64KのほかにVRAMを何10Kバイトも持てるのである。

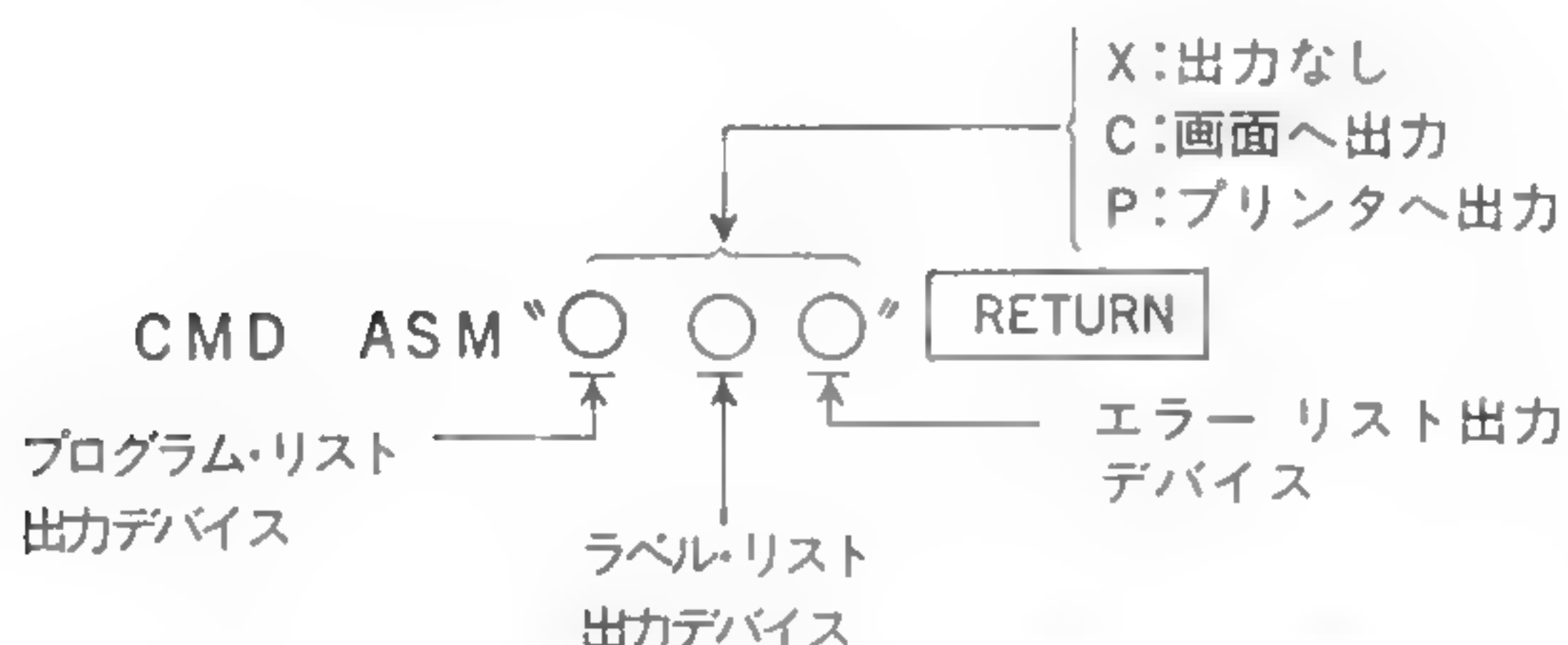
⑧MSXにはVRAMの使いかたのモードが何通りかあって、40×24のテキスト・モードはその1つ。1つのモードにしておいてVRAMの空いたところをアセンブラで利用しようとしているので、スクリーン・モードを変えるとVRAMのメモリ・マッ

プが変わってしまってアセンブラが利用できなくなってしまう。

⑨プログラムにある仕事をさせようとするときの、下準備のこと。ここでは、①スクリーン・モードの設定、②CLEAR文でBASICが使うメモリ範囲の限定、③同様にBASICが扱うファイル数の限定、などをしてからBLOADでマシン語のモニタ・アセンブラをロードしている。

1. アセンブラの起動

アセンブラはBASICのコマンド入力待ちの状態のとき、次のようなコマンドを入力することにより起動されます。



注) エラーリスト出力デバイスにXを指定しても画面にエラーを出力します。

このコマンドのうち、ダブル・クォーテーション (" ") で囲まれた3文字は、アセンブラが出力するプログラム・リスト⁽⁹⁰⁾、ラベル・リスト⁽⁹¹⁾、エラーリスト⁽⁹²⁾の出力を制御するためのものです。これらは省略することができます。もし全部省略すると、

CMD ASM RETURN

となり、プログラム・リスト、ラベル・リスト、エラーリストは画面に出力されます。

アセンブラがプログラム・リスト、ラベル・リストを出力しているとき、一時停止したければスペース・キーを押すことにより停止します。もう1度スペース・キーを押せば出力が再開されます。また、CTRL+STOPキーを入力すればアセンブルは中止され、アセンブラから抜け出してBASICのコマンド入力待ちの状態に戻ります。

2. モニタの起動

モニタはBASICのコマンド入力待ちのとき、

CMD MON RETURN

と入力することによって起動されます。

アセンブラを切り離してモニタだけを使うには、次の操作によりEC03番地から実行します。

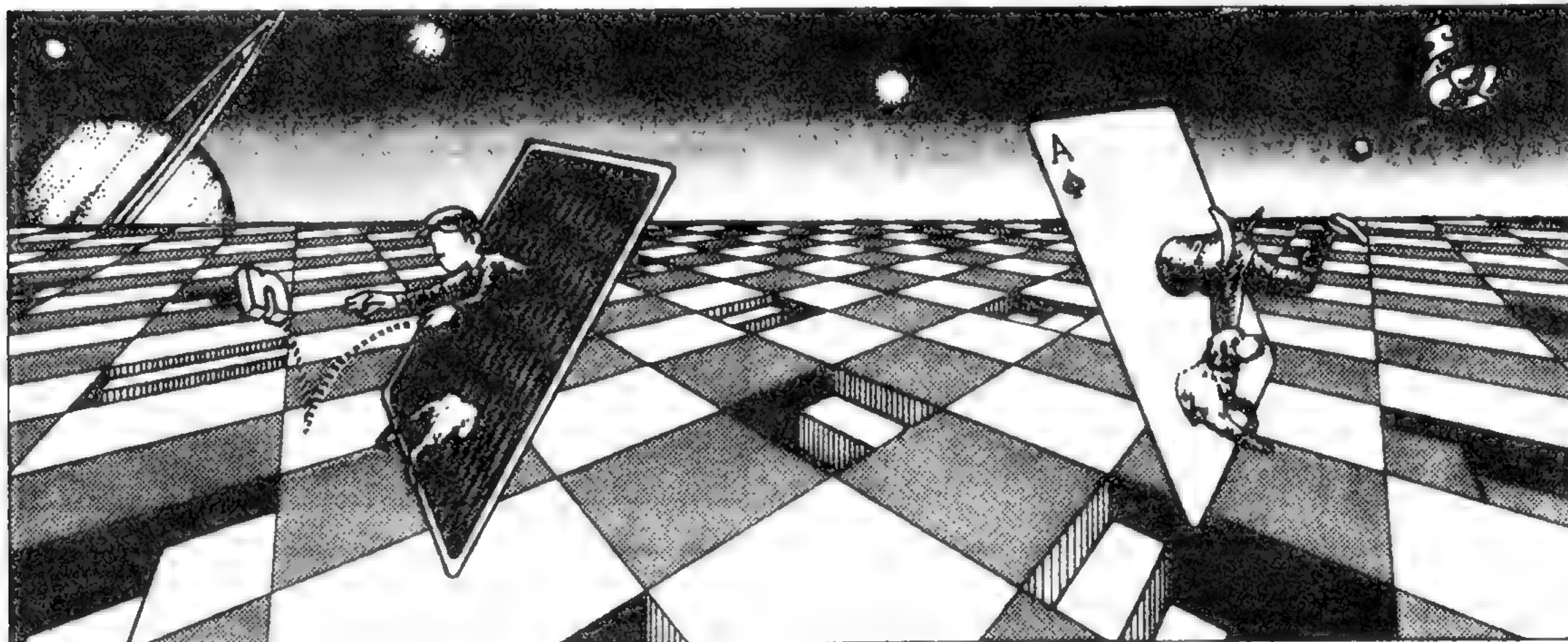
DEFUSR=&HEC03 RETURN

A=USR(0) RETURN

こうしてアセンブラを切り離したあと、モニタは、

CMD RETURN

だけで起動できます。



⁽⁹⁰⁾ アセンブルするときに出すモニタのリストのこと。モニタだけでなく、メモリに配置されときのアドレスやオブジェクトも一緒に出力される。アセンブル・リストとも呼ばれる。

⁽⁹¹⁾ アセンブルした全リスト中で使われたラベル名、ラベルの値を出力したリスト。

⁽⁹²⁾ アセンブル中に発見された種々のエラーを表示するリスト。エラーの位置、種類が出力される。

3

4 モニタ・アセンブラ

ストリング・サーチの使用法

ストリング・サーチの起動は、BASICのコマンド入力待ちで次のようにします。

CMD S “文字列” RETURN

ストリング・サーチが起動されると、ソース・プログラムの中から指定された文字列を含む行を探し、該当する行の行番号をすべて画面に出力します。文字列にダブル・クォーテーション(“ ”)を含むことはできません。

操作例4-1は第1章で使用したサンプル・プログラムのソース・プログラム中から「LD」という文字列を含む行を探したものです。

操作例 4-1 ストリング・サーチ

CMD S“LD”)

Assembler source string search

1080	1230	1240	1260
1270	1300	1340	1360
1370	1390	1400	1420
1430	1460	1510	1530
1540	1550	1560	1580
1590	1610	1620	1630
1640	1660	1670	1690
1700	1710	1720	1740
1750	1770	1780	1790
1800	1870	1920	1990
2010	2040	2090	2110
2190	2270	2310	

注) 下線が引かれている文字がキーボードからの入力です。
はリターン・キーのことです。

4

4 モニタ・アセンブラ

マシン語モニタの使用法

モニタを起動すると画面は、下のようになり、モニタはコマンド入力待ちになります。モニタのコマンドには表4-1のようなものがあります。

モニタ起動時の画面

OK
CMD MON↵

MSX Monitor Rev 1.0



カーソル

注) 下線が引かれている文字がキーボードからの入力です。
はリターン・キーのことです。

表4-1 モニタ・コマンド表

コマンド名	入力形式	内 容
Dコマンド 及び DSコマンド	[L]D[S](開始アドレス)[, 終了アドレス] RETURN	メモリ・ダンプ。先頭に“L”があると、プリンタに出力。“S”が付くとチェック・サム付きダンプ
Sコマンド	Sアドレス RETURN	メモリの内容表示と変更
Xコマンド	X(レジスタ名) RETURN	レジスタの内容表示と変更
Gコマンド	G(実行アドレス)[, ブレーク・ポイント1][, ブレーク・ポイント2] RETURN	マシン語プログラムの実行。ブレーク・ポイントは2個まで設定可能
Rコマンド	R(オフセット) RETURN	VRAMからオブジェクトをロード
Bコマンド	B RETURN	BASICに戻る

注) []内は省略可能

⑨デバッグ中に、たとえば1234Hから始まるサブルーチンに誤りが発見され、“CALL 1234H”というところをすべてほかの命令に置きかえる必要が生じたでしょう。このとき、全体が長いプログラムであればあるほど、全リストを出力して手作業で1つ1つ置きかえるのは時間と紙(と労力)の無駄遣いである。ストリング・サーチコマンドを利用すれば、

CMD S “CALL 1234H”

と打ち込むだけでプログラム中の“CALL 1234H”をまちがいなく残らず探し出してくれる。

CMD S “1234H”

と指定すれば、CALL命令だけでなく1234Hを参照している命令すべてを選び出すことができる。

入力したコマンドの訂正にはBSキーまたはESCキーを使います。BSキーは最後に入力された文字を消し、カーソルを1文字分左へ戻します。ESCキーは、入力した1行分をすべて消去します。ESCキーの代わりに[CTRL]+[C]も使えます。また、[SHIFT]+[CLS]を入力すれば、画面はクリアされます。

たとえば、「DEA00」と入力したところ、正しくは「DEC00」であったことに気づいたとしましょう。

・BSキーを使う場合

- * DEA00 (左の■はカーソル)
- * DEA0 ■ BSキーを1回押す。1文字消える。
- * DE ■ BSキーをあと2回押す。2文字消える。
- * DEC00 ■ C00とキーを押す。

「DEC00」に修正された。

・ESCキーを使う場合

- * DEA00 ■
- * ■ ESCキーを押す。「DEA00」が消える。
- * DEC00 ■ DEC00とキーを押す。

「DEC00」に修正された。

モニタのコマンド入力ではBASICのようなカーソル・エディット^④は使えませんので注意してください。

これよりモニタの各コマンド別の、入力形式と動作について説明します。[]で囲まれている内容は省略可能です。

● メモリ・ダンプ(DまたはDSコマンド)

● [L] D [開始アドレス] [, 終了アドレス]

[RETURN]

開始アドレスから終了アドレスまでのメモリ

の内容を16進数とASCIIコードでダンプします。先頭に“L”がある場合、プリンタに出力します。開始アドレスが省略されている場合、直前に出力したアドレスの次を開始アドレスとします。終了アドレスが省略されている場合、開始アドレスから128バイト分(8バイト×16行)出力します。開始アドレス/終了アドレスの指定は4桁以内の16進数を使います。5桁以上入力した場合は、右端から4桁を目的のアドレスとします。

操作例 4-2 ASCIIコード付きメモリ・ダンプ(Dコマンド)

LDEC00, EC7F [RETURN] を実行

```

EC00 C3 22 EC F3 01 05 00 11 テ"●月....
EC08 0D FE 21 17 EC ED B0 21 .※!..●コー!
EC10 FF EB 22 7A F3 18 0B F1 .+ "z月..円
EC18 C3 1C EC 00 E5 CD 22 EC テ.●.▲/"●
EC20 E1 C9 F3 ED 73 3E F3 31 1/月os>月1
EC28 3E F3 FD E5 DD E5 E5 D5 >月11111111
EC30 C5 F5 21 00 EC 22 40 F3 ナ時!..●"●月
EC38 3A 9A F2 B7 C2 ED EF 32 : 1年*110~2
EC40 9D F2 3D 32 9A F2 2A 3E 1年=2 1年*>
EC48 F3 22 9B F2 FB 21 56 EC 月"1年町!V●
EC50 CD 7D F1 F3 18 17 0D 0A 11円月....
EC58 4D 53 58 20 4D 6F 6E 69 MSX Moni
EC60 74 6F 72 20 20 52 65 76 tor Rev
EC68 20 31 2E 30 00 2A 38 F3 1.0.*8月
EC70 22 CA F2 01 05 00 11 DD "1年....ン
EC78 F2 21 E4 FE ED B0 3E C3 年!▲●コー>テ

```

アドレス メモリの内容の16進数
ダンプ。1行8バイト出
力されます。

メモリの内容をASCII
コードとして出力。00
H~1FH, 7FH, 0FFH
のコードはピリオド(.)
に変換されます。

注) MSX用プリンタでないでASCIIコードの部分は画面に出力したときと多少異なります。

上の操作例4-2は、EC00番地からEC7F番地までのメモリ・ダンプをプリンタに出力した例です。使ったプリンタがMSX用のプリンタでないために、ASCIIコード部分の文字が画面にダンプしたときと多少異なっています。

メモリ・ダンプを画面に出力した場合(先頭の“L”を省略)、画面の横の文字数が36以下だとASCIIコード部分の出力は行なわれま

④スクリーン・エディットともいう。画面上に表示されてさえいれば、いつでも加筆、訂正できるという機能。BASICのエディタはこの方式が一般的であるが、モニタのコマンド入力などには普及していない。

せん。

ダンプ中にスペース・キーを押すと一時停止し、再度スペース・キーを押すと続行します。RETURNキーを押すと中断し、モニタはコマンド入力待ちの状態になります。

●〔L〕DS〔開始アドレス〕〔, 終了アドレス〕

RETURN

開始アドレスから終了アドレスまでのメモリの内容をチェック・サムつき16進数でダンプします。ほかはDコマンドと同じです。

操作例 4-3 チェック・サム付きメモリ・ダンプ(DSコマンド)

LDSEC00 RETURN を実行

EC00	C3	22	EC	F3	01	05	00	11	:	DB
EC08	0D	FE	21	17	EC	ED	80	21	:	ED
EC10	FF	EB	22	7A	F3	18	0B	F1	:	8D
EC18	C3	1C	EC	00	E5	CD	22	EC	:	8B
EC20	E1	C9	F3	ED	73	3E	F3	31	:	5F
EC28	3E	F3	FD	E5	DD	E5	E5	D5	:	8F
EC30	C5	F5	21	00	EC	22	40	F3	:	1C
EC38	3A	9A	F2	B7	C2	ED	EF	32	:	4D
EC40	9D	F2	3D	32	9A	F2	2A	3E	:	F2
EC48	F3	22	9B	F2	FB	21	56	EC	:	00
EC50	CD	7D	F1	F3	18	17	0D	0A	:	74
EC58	4D	53	58	20	4D	6F	6E	69	:	AB
EC60	74	6F	72	20	20	52	65	76	:	C2
EC68	20	31	2E	30	00	2A	38	F3	:	04
EC70	22	CA	F2	01	05	00	11	DD	:	D2
EC78	F2	21	E4	FE	ED	B0	3E	C3	:	93

アドレス メモリの内容の16進数ダンプ。 チェック・サム
1行8バイト出力されます。

上の操作例4-3は、EC00番地から、128バイト分のチェック・サムつきダンプをプリンタに出力した例です。

チェック・サムの計算は次のようにしています。

アドレス	データ	チェック・サム
EC00	C3 22 EC F3 01 05 00 11	: DB

$0C3H + 22H + 0ECH + 0F3H + 01H + 05H + 00H + 11H = 2DBH$

チェック・サムの計算法にはいろいろな方法がありますが、このモニタでは1行で表わされる8バイト分のデータを合計し、下位8ビット2桁の16進数をチェック・サムとしています。

2 メモリの内容表示, 変更(Sコマンド)

●S アドレス RETURN

メモリの内容を変更するときに使います。アドレスは4桁以内の16進数で入力します。5桁以上入力した場合は、右端から4桁を目的のアドレスとします。

下の操作例4-4にSコマンドの使用例を示します。この例ではメモリのD300番地から7バイトを30H, 31H, 32H, 33H, 34H, 35H, 36Hに変更しています。

操作例 4-4 Sコマンドの使用例

*SD300

D300	01	30	←変更するデータを16進数で入力
D301	FF	31	
D302	FF	33	←キーの入力でアドレスが1つデクリメントされる
D303	FF	32	
D302	33	32	←変更するデータを16進数で入力
D303	FF	33	
D304	FF	34	←変更するデータを16進数で入力
D305	FF	35	
D306	FF	36	←変更するデータを16進数で入力
D307	FF		
D308	FF		←リターン・キーのみではメモリの内容は変更されない
D309	FF		
D30A	FF		←ピリオドを入力するとモニタはコマンド入力待ちの状態になります。
D30B	FF		

*

注) 下線が引かれている文字がキーボードからの入力です、\はリターン・キーのことです。

3 レジスタの内容表示, 変更(Xコマンド)

●X RETURN

レジスタ、フラグの内容を表示します。ただし、AF', BC', DE', HL'の内容は表示しません。P78の操作例4-5はXコマンドによるレジスタの表示例です。

●X レジスタ名 RETURN

指定されたレジスタ名の内容を表示、変更することができます。指定できるレジスタ名はA, F, B, C, D, E, H, L, AF, BC, DE, HL, IX,

IY, SP, PCです。

操作例4-6はレジスタ内容の変更例です。

操作例 4-5 Xコマンドによるレジスタの表示

```
*X↓
                SZ H FNC
A =7C      F =37(00110111)
BC=7C34  DE=39DB  HL=D4FF
IX=F2A3  IY=8002  SP=D1B9  PC=EC00
```

注) 下線が引かれている文字がキーボードからの入力です。↵はリターン・キーのことです。

操作例 4-6 Xコマンドによるレジスタの内容の変更

```
*X↓
                SZ H FNC
A =7C      F =37(00110111)
BC=7C34  DE=39DB  HL=D4FF
IX=F2A5  IY=00D5  SP=D1B9  PC=EC00
```

```
*XA↓
A =7C  12↓
*XF↓
F =37  1↓
*XB↓
B =7C  A↓
*XHL↓
HL=D4FF  D300↓
*XIY↓
IY=00D5  D400↓
*X↓
```

```
                SZ H PNC
A =12      F =01(00000001)
BC=0A34  DE=39DB  HL=D300
IX=F2A5  IY=D400  SP=D1B9  PC=EC00
```

注) 下線が引かれている文字がキーボードからの入力です。↵はリターン・キーのことです。

で設定できます。実行させたマシン語プログラムは実行中にブレーク・ポイントに達すると、レジスタとフラグの内容を表示しコマンド入力待ちの状態になります。

実行アドレスが省略されている場合は、プログラム・カウンタPCが示すアドレスから実行されます。

ブレーク・ポイントはROM上のマシン語プログラムには設定できませんので注意してください。

実行アドレス、ブレーク・ポイント1, 2は4桁以内の16進数で入力します。5桁以上入力すると右端から4桁を目的のアドレスとします。

P79の操作例4-7はブレーク・ポイントを設定した場合の実行例です。

ブレーク・ポイントを設定しないでマシン語プログラムを実行すると、「JP 0EC00H」という命令(マシン語では「C3, 00, EC」)をプログラムが実行しない限りモニタには戻ってきません。また、正しくブレーク・ポイントを設定しないで実行した場合もこのようなことになります。P79の図4-3は、いくつかのブレーク・ポイントの設定法を述べたものです。

4 マシン語プログラムの実行(Gコマンド)

●G[実行アドレス][, ブレーク・ポイント1]

[, ブレーク・ポイント2] RETURN

開始アドレスからマシン語プログラムを実行します。このとき、ブレーク・ポイントを2個ま

5 オブジェクトのロード(Rコマンド)

●R[オフセット] RETURN

アセンブラがVRAMに出力したオブジェクトをメイン・メモリにロードします。ロードでき

●ブレーク・ポイントによってブレークしたアドレスより実行させるときなどに使う。

●通常、オブジェクトがロードされるエリアはソース・プログラム中のORG 擬似命令で指定される。すなわち、ソース中で“ORG 8000H”と指定があればオブジェクトは8000番地からつくられ、ロードも8000番地から行なわれる。しかしRコマンドでオフセットを指定するとORGで示されたアドレス以外

のエリアにもロードできる。たとえば上の例でオフセットに1000Hを指定すると、9000Hからロードされる。もちろん、8000Hからアセンブルされたオブジェクトであるから9000番地に置いて実行できない。オフセット機能は、オブジェクトを本来置きたいメモリに何か別の大切なプログラム(たとえばアセンブラ)があって、とりあえず別のエリアに置いてしまおうというようなときに用いられる。

るメモリ・エリアは、BASICのCLEAR文で指定した上限アドレスからアセンブラが格納されているアドレスの手前（アセンブラを切り離しモニタだけにした場合は、モニタが格納されているアドレスの手前）までです（図4-4）。

オフセットは4桁以内の16進数で入力します。5桁以上入力した場合、右端から4桁をオフセットとします。オフセットを省略するとゼロを

指定したのと同じことになります。オフセットにより、オブジェクトはソース・プログラムで指定されたアドレス以外のメモリ・エリアにロードすることができます。⁽⁹⁾

操作例4-8は、Rコマンドの実行例です。

6 BASICへ戻る（Bコマンド）

● B RETURN

モニタからBASICに戻ります。

操作例 4-7 Gコマンドの実行例

```
*DD300,D307)
D300 21 34 12 00 00 00 00 00 !4.....
*GD300,D303)
Break D303

SZ H FNC
A =7C F =37(00110111)
BC=7C34 DE=39DB HL=1234
IX=F2A5 IY=00D5 SP=D1B9 PC=D303
```

実行するプログラムを確認

D303番地にブレーク・ポイントを設定し、D300番地より実行する。D303番地でブレークするとレジスタ、フラグの内容を表示してコマンド入力待ち状態に戻る。

注) 下線が引かれている文字がキーボードからの入力です。↵はリターン・キーのことです。

図 4-3 ブレーク・ポイントの設定法

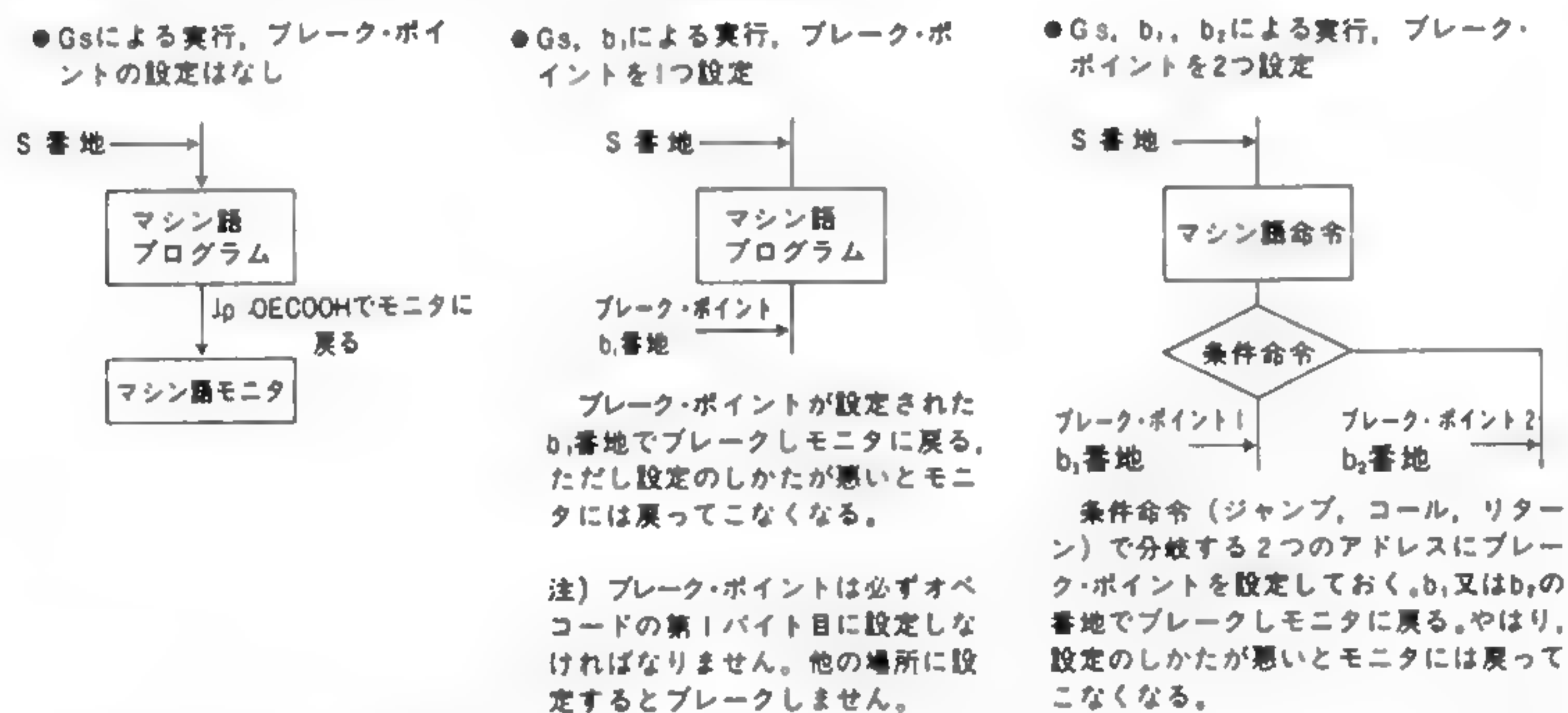
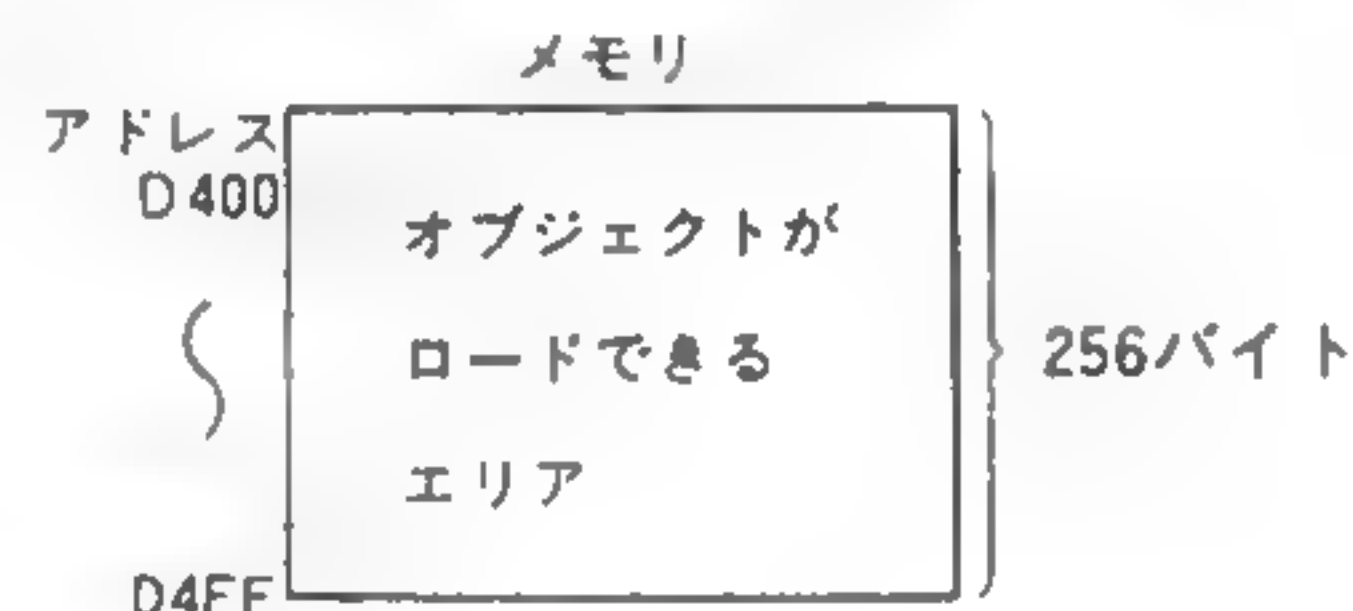


図 4-4 Rコマンドによりロードできるメモリ・エリア

例えばBASICで
CLEAR 0, &HD400
と実行していたとすると、

●アセンブラを切り離していない状態



操作例 4-8 Rコマンドの実行例

同じオブジェクトをオフセットを変えて別のアドレスへロードしてみる

```
*R↓
Free ( D200-D4FF ) } オフセットなし（ゼロ）でオブジェクトをロード。
                     } Freeはオブジェクトがロードできるエリアの上下限のアドレス、この
                     } 例ではD200番地からD4FF番地まで、Loadは実際にオブジェクトをロ
                     } ードしたアドレス、この例ではD300番地からD308-1番地まで。

Load D300-D308
*R100↓
Free ( D200-D4FF ) } オフセットを+100Hでオブジェクトをロード。
                     } 実際にロードしたアドレスは+100Hされたアドレス、この例ではD400
                     } 番地からD408-1番地まで。

Load D400-D408
*RFF00↓
Free ( D200-D4FF ) } オフセットを-100H (0-100H=0FF 00H) でオブジェクトをロード
                     } 実際にロードしたアドレスは-100Hされたアドレス、この例ではD200
                     } 番地からD208-1番地まで。

Load D200-D208
```

注) 下線が引かれている文字がキーボードからの入力です。↵はリターン・キーのことです。

5

4 モニタ・アセンブラ

セルフ・アセンブラの説明

ここで説明するMSX用のセルフ・アセンブラは、「2パス形式」と呼ばれるアセンブラです。「2パス形式」とは、ソース・プログラムを2回読み込んでオブジェクトをつくる方式です。1回目の読み込みでラベル・テーブル⁹⁷⁾をVRAM上に作成します。これを「パス1」と呼びます。2回目の読み込みでVRAM上にオブジェクトを出力します。これを「パス2」と呼びます。

アセンブラを動作させると、画面は次のようになります。

アセンブラ実行時の画面

```
OK
CMD ASM"XXC"

MSX Self Assembler Rev 1.0

Pass 1
No Error(s)

Pass 2
No Error(s)

End Address D454 . 38 Label(s)

OK
注) 下線が引かれている文字がキーボードからの入力です。
     \はリターン・キーのことです。
```

作成します。1行の構成は次のようになっています。

行番号△' [△] [ラベル:] [[△] [ニモニック]
△ [オペランド]] [△] [;コメント]

上記のように、ソース・プログラムはBASICのコメント行として記述します（行番号の直後に、引用符を付ける）。セーブ、ロードもBASICのプログラムと同様にできます。ここで〔 〕で囲まれた内容は省略できることを示します。ただし、オペランドはニモニックによっては書かないこともあります。ラベル、ニモニック、オペランド（文字定数以外）の中にスペースを入れることはできません。“△”は1つ以上のスペースを書くことを示します。

擬似命令EQUだけは構成が一部異なっていて、ラベルの後のコロン（:）がスペースになります。

リスト4-1は第1章で使ったサンプル・プログラムのソース・プログラムの一部を示したものです。

1. ソース・プログラムの形式

1 文の構成

ソース・プログラムはBASICのエディタで



⁹⁷⁾ そのソース・プログラム中で使われたラベルの一覧表。アセンブラはアSEMBル中のパス1でこの表を作成し、パス2でこれを見てラベルの値を知る。

⁹⁸⁾ ASCIIコード。“ASCII(American Standard Code for Informa-

tion Interchange)”は、日本のJISに当たるアメリカの規格で、アルファベットを始めとする文字コードの標準が定められている。どの機種のマニュアルにも巻末にASCIIコード表が掲載されているはずである。

リスト 4-1 ソース・プログラムの例(一部)

```

1000 :
1010 : Sample Program No. 1
1020 : (Machine Language)
1030 :
1040 : ORG 0D300H
1050 :
1060 : ** MSX ROMOS ENTRY **
1070 :
1080 : LDIRVM EQU 005CH
1090 : CHGMOD EQU 005FH
1100 : INIGRP EQU 0072H
1110 : CALPAT EQU 0084H
1120 : CALATR EQU 0087H
1130 : BREAKX EQU 0087H
1140 :
1150 : ** BASIC WORK ADDRESS **
1160 :
1170 : RG1SAV EQU 0F3E0H
1180 : SCRMOD EQU 0FCAFH
1190 :
1200 : ** MAIN **
1210 :
1220 : START:
1230 : LD A,(SCRMOD)
1240 : LD (SCMD),A
1250 :
1260 : LD HL,RG1SAV
1270 : LD A,(HL)
1280 : AND 0FCH
1290 : OR 2
1300 : LD (HL),A
1310 : CALL INIGRP
1320 :
1330 : XOR A
1340 : LD HL,SPRPT0
1350 : CALL SPRSET
1360 : LD A,1
1370 : LD HL,SPRPT1
1380 : CALL SPRSET
1390 : LD A,2
1400 : LD HL,SPRPT2
1410 : CALL SPRSET
1420 : LD A,3
1430 : LD HL,SPRPT3
1440 : CALL SPRSET
1450 :
1460 : LD HL,10000
1470 : LOOP:
1480 : PUSH HL
1490 :
1500 : XOR A
1510 : LD HL,SPTAT0
1520 : CALL SPRMOV
1530 : LD B,28
1540 : LD C,212+1
1550 : LD HL,X0
1560 : LD DE,D0
1570 : CALL POSUPD
1580 : LD A,1
1590 : LD HL,SPTAT1

```

2 文字

ソース・プログラムには、文字コードが⁽⁹⁸⁾00H～1FH, 7FHの文字以外で、キーボードから入力できて画面にも表示できる文字なら使えます。

3 ラベル

ラベルは英数字および“?”, “@”で構成された文字列です。ただし、先頭の文字に数字を使うことはできません。ラベルは先頭の6文字が有効で7文字以降は無視されます。ラベルに含まれる英小文字は英大文字と同等に処理されます。

ラベルにニモニック、レジスタ名、条件名を使うことはできません。

例) ・正しいラベルの例

ABC

? X436AYCN……初めの6文字 (? X436 Aまで) がラベルとなる

ADD1

@3921

ai3k……英大文字(AI3K)と同等に処理される

・誤ったラベルの例

1A3X ……数字で始まっている

@3 AS ……ラベル中にスペースがある

A ……レジスタ名は使えない

LD ……ニモニックはラベルに使えない

ラベルの定義はアドレスと定数では記述のしかたが異なります。アドレスを表わすラベルは、ラベル名の後にコロン(:)をつけることで定義されます。定数を表わすラベルの定義は擬似命令EQUによって行ないます。

例) ABC : ……ラベルABCに現在のアドレスが割りつけられる。

XY EQU 32……ラベルXYに定数32が割りつけられる。

このアセンブラではラベルを512個まで定義することができます。

4. ニモニック

ニモニックはザイログ形式の命令および擬似命令を使えます。英小文字で記述してもかまいません。

①Z80A CPUの命令

命令には次の67種類があります。

ADC ADD AND BIT CALL CCF CP
CPD CPDR CPI CPIR CPL DAA DEC
DI DJNZ EI EX EXX HALT IM
IN INC IND INDR INI INIR JP
JR LD LDD LDDR LDI LDIR NEG
NOP OR OTDR OTIR OUT OUTD OUTI
POP PUSH RES RET RETI RETN RL
RLA RLC RLCA RLD RR RRA RRC
RRCA RRD RST SBC SCF SET SLA
SRA SRL SUB XOR

命令の内容およびニモニックとオペランドの関係は、第3章 **Z80Aのマシン語命令**を参照してください。

②擬似命令

擬似命令はアセンブラに対して指示を与えるための命令です。次の7種類があります。

ORG END EQU DEFB DEFM DEFW DEFS

i) O R G 命令 (ORiGin)

ロケーション・カウンタ⁹⁹⁾にオペランドの値を設定します。オペランドは1つだけです。

ソース・プログラムの先頭にO R G 命令がないと、アセンブラはロケーション・カウンタにゼロを設定します。

例) ORG 0D300H……ロケーション・カウンタにD300を設定。

ii) E N D 命令

アセンブラに対してソース・プログラムの終了を指示します。オペランドはありません。

ソース・プログラムの最後の行のE N D 命令は省略できます。

iii) E Q U 命令 (EQUate)

オペランドの値を持つラベルを定義します。

E Q U 命令だけは次の形式をしています。
行番号△' [△] ラベル△E Q U △式 [△]
[;コメント]

例) XY EQU 100…ラベルXYに100を割りつける。

? XY EQU XY +3…ラベル?XYにラベルX
Yの値+3を割りつける。

iv) D E F B 命令 (DEFine Byte)

メモリに8ビットのデータを設定します。オペランドはカンマ(,)で区切れば複数個記述できます。

例) DEFB 100, 'A', 2F9H…この命令により、
次の3バイトがオブジェクトとして⁽¹⁰⁰⁾
出力される。

↓

64, 41, F9

v) D E F M 命令 (DEFine Memory)

メモリに文字列のデータを設定します。文字列には**4章5-1 2 文字**の規則にしたがった文字を書きます。ただし、引用符(')は2つ続けて書くことで1つの引用符となります。オペランドは1個だけで、次の形

⁹⁹⁾ アセンブラがその時点でどこのメモリを見ているかを表わすポインタ。アセンブルの最初にORG 命令でロケーション・カウンタに値を設定しておいて、その番地からオブジェクトを生成するようにする。

¹⁰⁰⁾ この例の場合、100は16進数にされて64, "A" は文字コードがそのまま入って41, 2F9Hは下位2桁(8ビット)が取られてF9となっている。

¹⁰¹⁾ 常に16進数4桁で計算されるということ。16進数2桁どう

式をしています。

行番号△' [△] [ラベル:] DEF M△ '文
字列' [△] [; コメント]

例) DEF M 'String'... この命令により次の
6バイトがオブジェク
トとして出力される。

↓

53, 74, 72, 69, 6E, 67

vi) DEF W 命令 (DEFine Word)

メモリに16ビットのデータを設定します。
データは上位8ビットと下位8ビットが逆
にメモリに落とされます。オペランドは、
カンマ(,)で区切れれば複数個記述できます。

例) DEF W 100, 'AB', 2F9H... この命令によ
り次の6バイト
がオブジェクト
として出力される。

↓

64, 00, 42, 41, F9, 02

vii) DEF S 命令 (DEFine Storage)

メモリにオペランドで示された大きさの領
域を確保します。オペランドは1個だけです。

例) DEF S 100... この命令により、100バイト
分のエリアが確保される。

5 オペランド

ニモニックによってオペランドが必要なもの
と必要ないものがあります。オペランドが2個
以上必要な場合は、カンマ(,)で各オペラン
ドを区切ります。

オペランドには、式およびレジスタ名、これら
をカッコで囲んだもの、または条件名を書きます。

①式

式は次のような形式をしています。

[{ + | - | } 定数 | ラベル | \$ }

[{ + | - | } 定数 | ラベル | \$ }] ...

ここで、[] で囲まれた内容は省略して
もよいことを示し、{ ① | ② | ③ } は①、②ま
たは③のいずれかを書くことを示します。[]
...は、[] で囲まれた内容を繰り返して書
いてもよいことを表わします。

計算は常に16ビットで行なわれ、オーバーフ
ローは無視されます。結果として1バイトの
値が欲しい場合、16ビットのうちの⁽¹⁰¹⁾下位8
ビットが使われ⁽¹⁰²⁾上位8ビットは無視されます。

i) 定数

定数には数値定数と文字定数があります。
数値定数は数字(0 ~ 9)で始まる文字列で、
10進定数と16進定数の2種類があります。

・10進定数

データの最後に“D”をつけるか、何も
つけません。

nnnnn D または nnnnn

nは0 ~ 9の数字です。nnnnnには0 ~ 65535
までの値を記述することができます。Dは
小文字のdでもかまいません。

・16進定数

データの最後に“H”をつけます。

nnnnn H

nは0 ~ 9の数字とA ~ Fの英字です。初
めの文字がA ~ Fの英字のときは先頭に数
字の0をつけます。⁽¹⁰³⁾nnnnnには0 ~ 0FFFF
までの16進数の値を記述します。A ~ F、
Hは英小文字でもかまいません。

しの計算でも、たとえば0012H + 0034H = 0046Hのように4
桁で計算する。

⑩たとえば00ABH + 00CDH = 0178Hだが、結果には78Hだ
けが使われるということ。

⑪アルファベットで始まる文字列はラベルと見分けられない
ため。

・文字定数

文字定数は、1個または2個の文字を引用符(')で囲んだものです。**4章5-1 2 文字**の規則にしたがっていればどのような文字でも書くことができます。ただし、引用符(')は2つ続けて書くことで1つの引用符となります。式の中では3文字以上の文字定数は使えません。

ii) ラベル参照

ラベルを式の中で使う場合、ラベルは必ず定義されていなければなりません。もし定義されていないラベルを参照するとエラーになります。

iii) \$ (ロケーション・カウンタ)

\$ は、\$ をオペランドに記述した命令の1バイト目のロケーション・カウンタの値を示します。

たとえばD308番地に「LD HL, \$」という命令があったとするとオブジェクトは次のようになります。

アドレス	命 令	オブジェクト
D 3 0 8	LD HL, \$	→2108D3

2 レジスタ名と条件名

レジスタ名と条件名には次の23種類のものがあります。これらは英小文字でもかまいません。

A B C D E H L I R AF BC DE HL IX

IY SP NC Z NZ P M PE PO

6 コメント

コメントとはセミコロン (;) の後の文字から行の終りまでの文字列のことで、**4章5-1 2 文字**の規則にしたがっていればどのような文字でも書くことができます。アセンブラはコメントを無視します。

2. プログラム・リストとラベル・リスト

リスト4-2がプログラム・リストのプリンタへの出力例です。画面に出力した場合、1行目のタイトルとページ番号、各行の行番号は出てきません。

プログラム・リストを見る上で注意しなければならないことは、オブジェクトが初めの4バイトしか出力されないということです。したがって、DEFB, DEFW, DEFM命令等のオブジェクトは全部出力されるとは限りません。たとえば「DEFB 1,2,3,4,5,6」という命令は、実際には6バイトのオブジェクト (01,02,03,04,05,06) が生成されますが、プログラム・リスト上には4バイト分(01,02,03,04)しか出力しません。

P84のリスト4-3がラベル・リストのプリンタへの出力例です。画面に出力した場合、1行に1つのラベルを出力します。

リスト 4-3 ラベル・リストの例

00B7	BREHL	D393	BREND	0087	CALATR	0084	CALPAT
005F	CHGMOD	D440	D0	D441	D1	D442	D2
D443	D3	0072	INIGRP	005C	LDIRVM	D334	LOOP
D3A4	POSUP1	D3A8	POSUP2	D39A	POSUPD	F3E0	RG1SAV
D43F	SCMD	FCAF	SCRMOD	D3AA	SPRMOV	D3BF	SPRPT0
D3DF	SPRPT1	D3FF	SPRPT2	D41F	SPRPT3	D3B3	SPRSET
D444	SPTAT0	D448	SPTAT1	D44C	SPTAT2	D450	SPTAT3
D300	START	D3BA	VRMOV	D445	X0	D449	X1
D44D	X2	D451	X3	D444	Y0	D448	Y1
D44C	Y2	D450	Y3				

注) プリンタへラベル・リストを出力すると1行に4つラベルが出力される。

リスト 4-2 プログラム・リストの例

MSX Self Assembler Rev 1.0 PAGE 1

```

1000:      ;
1010:      ; Sample Program No. 1
1020:      ;   (Machine Language)
1030:      ;
1040:      D300      ORG 0D300H
1050:      ;
1060:      ; ** MSX ROMOS ENTRY **
1070:      ;
1080:      005C =     LDIRVM EQU 005CH
1090:      005F =     CHGMOD EQU 005FH
1100:      0072 =     INIGRP EQU 0072H
1110:      0084 =     CALPAT EQU 0084H
1120:      0087 =     CALATR EQU 0087H
1130:      00B7 =     BREAKX EQU 00B7H
1140:      ;
1150:      ; ** BASIC WORK ADDRESS **
1160:      ;
1170:      F3E0 =     RG1SAV EQU 0F3E0H
1180:      FCAF =     SCRMOD EQU 0FCAFH
1190:      ;
1200:      ; ** MAIN **
1210:      ;
1220:      D300      START:
1230:      D300 3AAFFC      LD A,(SCRMOD)
1240:      D303 323FD4      LD (SCMD),A
1250:
1260:      D306 21E0F3      LD HL,RG1SAV
1270:      D309 7E          LD A,(HL)
1280:      D30A E6FC        AND 0FCH
1290:      D30C F602        OR 2
1300:      D30E 77          LD (HL),A
1310:      D30F CD7200      CALL INIGRP
1320:      ;
1330:      D312 AF          XOR A
1340:      D313 21BFD3      LD HL,SPRPT0
1350:      D316 CDB3D3      CALL SPRSET
1360:      D319 3E01        LD A,1
1370:      D31B 21DFD3      LD HL,SPRPT1
1380:      D31E CDB3D3      CALL SPRSET
1390:      D321 3E02        LD A,2
1400:      D323 21FFD3      LD HL,SPRPT2
1410:      D326 CDB3D3      CALL SPRSET
1420:      D329 3E03        LD A,3
1430:      D32B 211FD4      LD HL,SPRPT3
1440:      D32E CDB3D3      CALL SPRSET
1450:      ;
1460:      D331 211027      LD HL,10000
1470:      D334      LOOP:
1480:      D334 E5          PUSH HL
1490:

```


3. エラーメッセージ

エラーメッセージは表4-2に示すように全部で15種類あります。そのうち、アセンブルを中止するエラーは4種類あります。他のエラーはエラーメッセージを表示するだけでアセンブルを続行します。ただし、パス1でエラーが見つかった場合、パス2は行なわれません。パス2でエラーが発生した場合、エラーリスト出力デバイスにエラーメッセージを出力するほか、プログラム・リストにもエラーコードが出力されます（リスト4-4、4-5）。

表4-2 アセンブラが出力するエラーメッセージ
(アセンブルを中止するもの)

メッセージ	意 味
% Object area full	オブジェクト・エリアがいっぱいになった。
% Label table full	ラベル・テーブルがいっぱいになった
% Assembler source error	ソースがアセンブラのソースでない
% Screen not 40×24 text mode	40×24のテキスト・モードでない

(アセンブルを中止しないもの)

エラーコード	メッセージ	意 味
A	Address Overflow	アドレスがFFFF番地を超えた
B	Balance Error	左右のカッコの数が合わない。引用符の使いかたがおかしい。
E	Expression Error	演算子または式の記述がおかしい。
F	Format Error	オペランドの数が合わない、該当するニモニックがない、その他
L	Label Error	ラベルに予約語を使っている。または記述がおかしい
M	Multiply Defined Label	同じラベル名を複数個定義した
O	Operand Error	オペランドの記述がおかしい
P	Phase Error	パス1とパス2でラベルの値が異なる
R	Reference Error	レラティブ・ジャンプが範囲外、インデックス・アドレッシングでディスプレイメントが範囲外
U	Undefined Label	未定義ラベルを参照した
V	Value Error	オペランドに記述された値が違っている

リスト 4-4 エラーリスト

この例はパス1でのエラーです。

```
1080: Label Error
1100: Format Error
1120: Operand Error
```

行番号 エラー メッセージ

リスト 4-5 ① パス2でのエラーコード

CMD ASM* PXC* によりプログラム・リストにエラーコードが出力された例

```
1000: :
1010: : Error Sample
1020: :
1030: EC00 = MONIT EQU 0EC00H
1040: 00A2 = CHRPUT EQU 00A2H
1050: :
1060: D400 ORG 0D400H
1070: U D400 210000 LD HL,MESG
1080: D403 LOOP:
1090: D403 7E LD A,(HL)
1100: D404 23 INC HL
1110: D405 B7 OR A
1120: D406 CA00EC JP Z,MONIT
1130: U D409 CD0000 CALL CHRUT
1140: D40C 06FF LD B,0FFH
1150: D40E 10FE DJNZ $
1160: D410 18F1 JR LOOP
1170: D412 53616D70 MSG:DEFM 'Sample'
1180: D418 00 DEFB 0
1190: D419 END
```

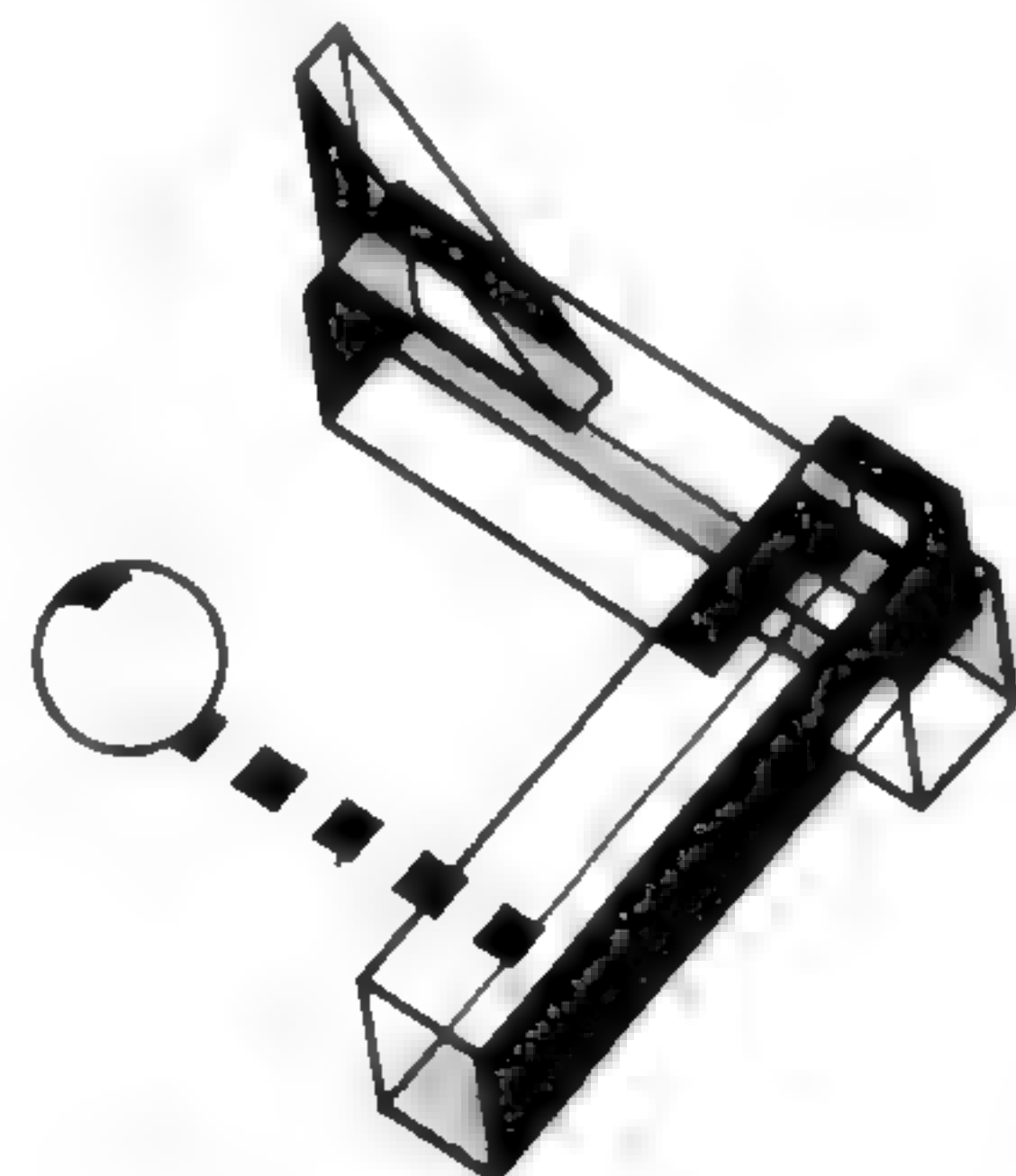
エラーコード

リスト 4-5 ② パス2でのエラーメッセージ

CMD ASM* PXP* によりプリンタへプログラム・リスト、エラーリストを出力した例

```
1000: :
1010: : Error Sample
1020: :
1030: EC00 = MONIT EQU 0EC00H
1040: 00A2 = CHRPUT EQU 00A2H
1050: :
1060: D400 ORG 0D400H
1070: Undefined Label
1070: U D400 210000 LD HL,MESG
1080: D403 LOOP:
1090: D403 7E LD A,(HL)
1100: D404 23 INC HL
1110: D405 B7 OR A
1120: D406 CA00EC JP Z,MONIT
1130: Undefined Label
1130: U D409 CD0000 CALL CHRUT
1140: D40C 06FF LD B,0FFH
1150: D40E 10FE DJNZ $
1160: D410 18F1 JR LOOP
1170: D412 53616D70 MSG:DEFM 'Sample'
1180: D418 00 DEFB 0
1190: D419 END
```

エラーメッセージ





モニタ・アセンブラの入力方法

まず、初めにリスト4-6のマシン語モニタを打ち込みます。打ち込み終わったら、必ずセーブしておいてください。次に、

SCREEN 0 RETURN

RUN RETURN

と入力します。リストが正しく打ち込まれていればマシン語モニタが起動します（この間多少時間がかかる）。もし暴走するようなら電源を入れなおし、セーブしておいたプログラムを再ロードしてミスを修正してください。モニタが起動できたら、Rコマンド以外の全コマンドを4章4 マシン語モニタの使用法に示す操作例を参考に動作チェックしてください（Rコマンドはアセンブラのチェックのとき一緒にチェックする）。

次にBASICに戻り（モニタのBコマンド）、

NEW RETURN

CLEAR 100, &HD500 RETURN

を行ない、

CMD RETURN

でモニタを起動します。そしてモニタのSコマンドでリスト4-7のセルフ・アセンブラを打ち込みます。リスト4-7はチェック・サムつきのダンブ・リストなので、打ち込むときにはデータ部⁽¹⁰⁴⁾だけを入力してください。

打ち込みが終わったら入力ミスがないことを確認して、

BSAVE "CAS:MSXASM", &HD500, &HF37F, &HD503 RETURN

でカセットへセーブしてください。あとは4章2 起動方法で説明した操作でモニタ・アセンブラが起動できます。

リスト4-8にZ80Aの全ニモニックと擬似命令のアセンブル・リストを示します。これで全ニモニックと擬似命令がアセンブルできるかどうかチェックしておいた方が（あとのためにも）よいでしょう。

リスト 4-6 マシン語モニタのプログラム・リスト

```
100
110 ' MSX Monitor
120 '   Rev 1.0
130 '
200 CLEAR 100, &HEC00
210 DEF USR=&HEC03
220 I=&HEC00
230 READ A$: IF A$="*" THEN 300
240 POKE I, VAL("&H"+A$): I=I+1
250 GOTO 230
300 A=USR(0)
310 END
1000 DATA C3,22,EC,F3,01,05,00,11
1010 DATA 0D,FE,21,17,EC,ED,B0,21
1020 DATA FF,EB,22,7A,F3,18,0B,F1
1030 DATA C3,1C,EC,00,E5,CD,22,EC
```

```
1040 DATA E1,C9,F3,ED,73,3E,F3,31
1050 DATA 3E,F3,FD,E5,DD,E5,E5,D5
1060 DATA C5,F5,21,00,EC,22,40,F3
1070 DATA 3A,9A,F2,B7,C2,ED,EF,32
1080 DATA 9D,F2,3D,32,9A,F2,2A,3E
1090 DATA F3,22,9B,F2,FB,21,56,EC
1100 DATA CD,7D,F1,F3,18,17,0D,0A
1110 DATA 4D,53,58,20,4D,6F,6E,69
1120 DATA 74,6F,72,20,20,52,65,76
1130 DATA 20,31,2E,30,00,2A,38,F3
1140 DATA 22,CA,F2,01,05,00,11,DD
1150 DATA F2,21,E4,FE,ED,B0,3E,C3
1160 DATA 21,9C,EF,32,E4,FE,22,E5
1170 DATA FE,31,32,F3,AF,32,9D,F2
1180 DATA 32,C9,F2,3E,DF,32,D7,F2
1190 DATA 32,DA,F2,FB,CD,74,F1,CD
```

⑩最も左の4桁がアドレス部、次の2桁×8がデータ部、右のコロン（:）の次の2桁がチェック・サム部。


```

1200 DATA 2C,ED,78,B7,28,E3,7E,23
1210 DATA FE,4C,20,08,3E,FF,32,9D
1220 DATA F2,7E,23,FE,44,20,4C,FE
1230 DATA 44,20,11,7E,FE,53,20,06
1240 DATA 23,3E,FF,32,C9,F2,22,9F
1250 DATA F2,C3,FA,ED,22,9F,F2,FE
1260 DATA 53,CA,E8,EE,FE,58,CA,25
1270 DATA F0,FE,47,CA,34,EF,FE,42
1280 DATA 28,08,FE,52,CA,B1,F1,2B
1290 DATA 22,9F,F2,18,16,F3,01,05
1300 DATA 00,11,E4,FE,21,DD,F2,ED
1310 DATA B0,AF,32,9A,F2,ED,7B,9B
1320 DATA F2,FB,C9,AF,32,9D,F2,CD
1330 DATA 74,F1,3E,3F,CD,A2,00,CD
1340 DATA 6A,F1,2A,9F,F2,7E,B7,28
1350 DATA 0A,FE,2C,28,06,CD,A2,00
1360 DATA 23,18,F2,CD,C0,00,C3,89
1370 DATA EC,AF,18,07,3E,2A,CD,A2
1380 DATA 00,3E,FF,32,9E,F2,CD,56
1390 DATA 01,06,00,21,A1,F2,22,9F
1400 DATA F2,CD,9F,00,FE,61,38,06
1410 DATA FE,7B,30,02,E6,5F,FE,0D
1420 DATA 28,6B,FE,1B,28,04,FE,03
1430 DATA 20,0D,78,B7,28,E3,3E,7F
1440 DATA CD,A2,00,10,FB,18,D2,FE
1450 DATA 08,20,0D,78,B7,28,D2,3E
1460 DATA 7F,CD,A2,00,05,2B,18,C9
1470 DATA FE,0C,20,0D,3A,9E,F2,B7
1480 DATA 28,BF,3E,0C,CD,A2,00,18
1490 DATA A3,FE,01,20,15,CD,9F,00
1500 DATA FE,41,38,0E,FE,60,38,A9
1510 DATA FE,61,38,A5,FE,7B,30,02
1520 DATA E6,5F,FE,20,38,9B,FE,7F
1530 DATA 28,97,4F,77,23,04,78,FE
1540 DATA 28,38,04,2B,05,0E,07,79
1550 DATA CD,A2,00,18,84,36,00,21
1560 DATA 41,F2,C9,DD,2A,9F,F2,0E
1570 DATA 00,61,69,DD,7E,00,B7,28
1580 DATA 23,FE,2C,28,1D,D6,30,FE
1590 DATA 0A,38,0C,D6,07,FE,0A,DA
1600 DATA 03,ED,FE,10,D2,03,ED,29
1610 DATA 29,29,29,B5,6F,0C,DD,23
1620 DATA 18,D9,DD,23,47,DD,22,9F
1630 DATA F2,C9,CD,C3,ED,79,B7,28
1640 DATA 03,22,CA,F2,78,B7,28,0F
1650 DATA CD,C3,ED,79,B7,CA,03,ED
1660 DATA 78,B7,C2,03,ED,18,07,2A
1670 DATA CA,F2,11,7F,00,19,22,CC
1680 DATA F2,3A,9D,F2,B7,C4,74,F1
1690 DATA 06,00,78,B7,20,13,DD,21
1700 DATA CE,F2,DD,36,00,00,CD,74
1710 DATA F1,2A,CA,F2,CD,86,F1,0E
1720 DATA 00,CD,6A,F1,2A,CA,F2,7E
1730 DATA CD,8B,F1,7E,81,4F,7E,FE
1740 DATA 20,38,08,FE,7F,28,04,FE
1750 DATA FF,20,02,3E,2E,DD,77,00
1760 DATA DD,23,DD,36,00,00,23,22
1770 DATA CA,F2,2B,04,ED,5B,CC,F2
1780 DATA B7,ED,52,30,2A,78,FE,08
1790 DATA 20,C7,CD,B4,EE,CD,9C,00

```

```

1800 DATA 28,A6,CD,9F,00,FE,0D,CA
1810 DATA 89,EC,CD,56,01,CD,9C,00
1820 DATA 28,FB,CD,9F,00,FE,0D,CA
1830 DATA 89,EC,CD,56,01,18,89,78
1840 DATA FE,08,28,0A,C5,06,03,CD
1850 DATA 6E,F1,C1,04,18,F1,CD,B4
1860 DATA EE,C3,89,EC,3A,C9,F2,B7
1870 DATA 28,0F,CD,6A,F1,3E,3A,CD
1880 DATA 9E,F1,CD,6A,F1,79,C3,8B
1890 DATA F1,3A,9D,F2,B7,20,10,3A
1900 DATA B0,F3,FE,25,D8,20,08,CD
1910 DATA DF,EE,3E,08,C3,A2,00,CD
1920 DATA 6A,F1,21,CE,F2,C3,7D,F1
1930 DATA CD,C3,ED,79,B7,CA,03,ED
1940 DATA 78,B7,C2,03,ED,22,CC,F2
1950 DATA CD,74,F1,CD,86,F1,CD,6A
1960 DATA F1,7E,CD,8B,F1,CD,6A,F1
1970 DATA CD,29,ED,78,B7,28,19,3D
1980 DATA 20,0A,7E,FE,2E,CA,89,EC
1990 DATA FE,5E,28,12,CD,C3,ED,B7
2000 DATA C2,03,ED,7D,2A,CC,F2,77
2010 DATA 2A,CC,F2,23,18,C7,2A,CC
2020 DATA F2,2B,18,C1,CD,C3,ED,ED
2030 DATA 5B,40,F3,79,B7,20,01,EB
2040 DATA 22,CC,F2,78,B7,28,2F,CD
2050 DATA C3,ED,79,B7,CA,03,ED,E5
2060 DATA 78,B7,28,18,CD,C3,ED,79
2070 DATA B7,CA,03,ED,E5,78,B7,C2
2080 DATA 03,ED,E1,22,DB,F2,7E,32
2090 DATA DA,F2,36,DF,E1,22,D8,F2
2100 DATA 7E,32,D7,F2,36,DF,2A,CC
2110 DATA F2,22,40,F3,F3,31,32,F3
2120 DATA ED,5B,40,F3,2A,3E,F3,2B
2130 DATA 72,2B,73,22,3E,F3,F1,C1
2140 DATA D1,E1,DD,E1,FD,E1,ED,7B
2150 DATA 3E,F3,FB,C9,E5,D5,21,08
2160 DATA 00,39,5E,23,56,1B,3A,D7
2170 DATA F2,FE,DF,28,17,2A,D8,F2
2180 DATA B7,ED,52,28,14,3A,DA,F2
2190 DATA FE,DF,28,08,2A,DB,F2,B7
2200 DATA ED,52,28,05,D1,E1,C3,DD
2210 DATA F2,F3,D1,E1,F1,F1,ED,73
2220 DATA 3E,F3,31,3E,F3,FD,E5,DD
2230 DATA E5,E5,D5,C5,F5,2A,3E,F3
2240 DATA 5E,23,56,23,22,3E,F3,1B
2250 DATA ED,53,40,F3,FB,3A,D7,F2
2260 DATA FE,DF,28,0F,2A,D8,F2,77
2270 DATA 3A,DA,F2,FE,DF,28,04,2A
2280 DATA DB,F2,77,AF,32,9D,F2,21
2290 DATA 1C,F0,CD,7D,F1,2A,40,F3
2300 DATA 22,CA,F2,CD,86,F1,CD,74
2310 DATA F1,C3,B4,F0,0D,0A,42,72
2320 DATA 65,61,6B,20,00,2A,9F,F2
2330 DATA 7E,B7,CA,B4,F0,1E,20,57
2340 DATA 23,7E,B7,28,07,5F,23,7E
2350 DATA B7,C2,03,ED,01,00,10,21
2360 DATA 3A,F1,7E,BA,23,20,04,7E
2370 DATA BB,28,08,23,23,0C,10,F2
2380 DATA C3,03,ED,2B,CD,74,F1,CD
2390 DATA 7D,F1,3E,3D,CD,A2,00,06

```



```

2400 DATA 00,79,FE,08,30,21,21,32
2410 DATA F3,09,7E,CD,8B,F1,E5,CD
2420 DATA 6A,F1,CD,29,ED,78,B7,CA
2430 DATA 89,EC,CD,C3,ED,B7,C2,03
2440 DATA ED,7D,E1,77,C3,89,EC,D6
2450 DATA 08,87,4F,21,32,F3,09,23
2460 DATA 7E,CD,8B,F1,2B,E5,7E,CD
2470 DATA 8B,F1,CD,6A,F1,CD,29,ED
2480 DATA 78,B7,CA,89,EC,CD,C3,ED
2490 DATA B7,C2,03,ED,EB,E1,73,23
2500 DATA 72,C3,89,EC,CD,74,F1,06
2510 DATA 0E,CD,6E,F1,21,31,F1,CD
2520 DATA 7D,F1,CD,74,F1,11,3D,F1
2530 DATA CD,0E,F1,2A,32,F3,E5,7C
2540 DATA CD,8B,F1,06,03,CD,6E,F1
2550 DATA 11,3A,F1,CD,0E,F1,E1,7D
2560 DATA CD,8B,F1,3E,28,CD,A2,00
2570 DATA 06,08,26,18,29,7C,CD,A2
2580 DATA 00,10,F7,3E,29,CD,A2,00
2590 DATA CD,74,F1,11,55,F1,21,34
2600 DATA F3,06,03,CD,19,F1,06,04
2610 DATA CD,19,F1,C3,89,EC,EB,CD
2620 DATA 7D,F1,EB,13,3E,3D,C3,A2
2630 DATA 00,CD,0E,F1,D5,5E,23,56
2640 DATA 23,EB,CD,86,F1,EB,D1,10
2650 DATA 03,C3,74,F1,CD,6A,F1,18
2660 DATA E8,53,5A,20,48,20,50,4E
2670 DATA 43,00,46,20,00,41,20,00
2680 DATA 43,20,00,42,20,00,45,20
2690 DATA 00,44,20,00,4C,20,00,48
2700 DATA 20,00,41,46,00,42,43,00
2710 DATA 44,45,00,48,4C,00,49,58
2720 DATA 00,49,59,00,53,50,00,50
2730 DATA 43,00,3E,20,18,30,CD,6A
2740 DATA F1,10,FB,C9,3E,0D,CD,9E
2750 DATA F1,3E,0A,18,21,7E,B7,C8
2760 DATA CD,9E,F1,23,18,F7,7C,CD
2770 DATA 8B,F1,7D,F5,0F,0F,0F,0F
2780 DATA CD,94,F1,F1,E6,0F,FE,0A
2790 DATA 38,02,C6,07,C6,30,F5,3A
2800 DATA 9D,F2,B7,20,04,F1,C3,A2
2810 DATA 00,F1,CD,A5,00,DA,89,EC
2820 DATA C9,CD,C3,ED,78,B7,C2,03
2830 DATA ED,E5,FD,E1,3A,AF,FC,B7
2840 DATA 20,32,21,EA,F1,CD,7D,F1
2850 DATA 2A,4A,FC,CD,86,F1,3E,2D
2860 DATA CD,9E,F1,2A,7A,F3,CD,86
2870 DATA F1,CD,6A,F1,3E,29,CD,9E
2880 DATA F1,CD,74,F1,CD,1A,F2,C3
2890 DATA 89,EC,0D,0A,46,72,65,65
2900 DATA 20,28,20,00,21,FD,F1,CD
2910 DATA 7D,F1,C3,89,EC,0D,0A,53
2920 DATA 63,72,65,65,6E,20,6E,6F
2930 DATA 74,20,34,30,58,32,34,20
2940 DATA 74,65,78,74,20,6D,6F,64
2950 DATA 65,00,21,00,20,18,0A,EB
2960 DATA 3E,2D,CD,9E,F1,CD,86,F1
2970 DATA EB,CD,4A,00,23,5F,CD,4A
2980 DATA 00,23,57,CD,4A,00,23,4F
2990 DATA CD,4A,00,23,47,B1,B2,B3

```

```

3000 DATA C8,E5,FD,E5,E1,19,EB,21
3010 DATA 75,F2,CD,7D,F1,EB,CD,86
3020 DATA F1,EB,E1,78,B1,28,C8,E5
3030 DATA 2A,4A,FC,2B,B7,ED,52,E1
3040 DATA 30,1B,E5,2A,7A,F3,B7,ED
3050 DATA 52,E1,38,11,CD,4A,00,12
3060 DATA 13,23,0B,18,DE,0D,0A,4C
3070 DATA 6F,61,64,20,00,CD,74,F1
3080 DATA EB,CD,86,F1,21,8D,F2,CD
3090 DATA 7D,F1,C3,89,EC,20,20,4C
3100 DATA 6F,61,64,20,65,72,72,6F
3110 DATA 72,00,00,D6,D0,00,FF,A2
3120 DATA F2,42,00,54,00,00,00,00
3130 DATA 00,00,00,00,00,00,00,00
3140 DATA 00,00,00,00,00,00,00,00
3150 DATA 00,00,00,00,00,00,00,00
3160 DATA 00,00,00,00,00,00,00,00
3170 DATA 00,00,FF,D4,00,00,00,00
3180 DATA 00,00,00,00,00,00,00,DF
3190 DATA 00,00,DF,00,00,C9,C9,C9
3200 DATA C9,C9,00,00,00,00,00,00
3210 DATA 00,00,00,00,00,00,00,00
3220 DATA 00,00,00,00,00,00,67,0F
3230 DATA 67,0F,A8,0D,01,00,3F,01
3240 DATA 7F,04,EC,FB,E1,FB,69,0D
3250 DATA F3,0C,04,00,02,00,00,00
3260 DATA 8A,2E,90,E9,04,46,A8,7F
3270 DATA FF,00,75,03,DE,F3,90,0D
3280 DATA 20,00,F8,0B,17,03,FD,10
3290 DATA 42,01,E2,F2,A2,F2,44,ED
3300 DATA A2,EC,37,7C,34,7C,DB,39
3310 DATA FF,D4,04,00,02,00,D6,D0
3320 DATA 00,EC,00,00,00,00,00,00
3330 DATA 00,00,00,00,00,00,00,00
3340 DATA 00,00,00,00,00,00,00,00
3350 DATA 00,00,00,00,00,00,00,00
3360 DATA 00,00,00,00,00,00,00,00
3370 DATA 00,00,00,00,00,00,50,7E
3380 DATA 64,01,C8,00,CF,7C,00,00
3390 DATA 00,00,FF,D4,00,00,00,00
3400 DATA *

```

このモニタ・アセンブラとグラフィック・エディタを1本のカセットに収め、全国の書店・ショップなどで発売しております。打ち込むのはたいへん時間のかかる作業なので、面倒な方はご利用ください。

「MONITOR ASSEMBLER/GRAPHIC EDITOR」 ¥4,800(MIA)

また、小社発行の「快速マシン語ゲーム集」に掲載されたモニタ・プログラムに比べて若干データが異なりますが、動作に変更はありません。

リスト 4-7 セルフ・アセンブラのダンプ・リスト

D500	C3	2B	D5	C3	18	D5	C3	E5	:	1B	D6D0	FE	3A	C8	FE	20	C0	18	F4	:	EA
D508	D6	C3	2E	D7	C3	96	D9	C3	:	93	D6D8	32	7D	F3	78	32	7E	F3	79	:	36
D510	8C	D9	C3	25	D5	C3	25	D5	:	DF	D6E0	32	7F	F3	AF	01	3E	01	32	:	C5
D518	F3	01	05	00	11	0D	FE	21	:	36	D6E8	7C	F3	ED	73	86	EB	31	70	:	E1
D520	26	D5	ED	B0	FB	C9	C3	00	:	1F	D6F0	F3	21	88	EB	01	25	00	36	:	E3
D528	D5	00	00	7E	FE	4D	28	0A	:	D0	D6F8	00	23	0B	78	B1	20	F8	3D	:	AC
D530	FE	41	28	1C	FE	53	CA	B5	:	53	D700	3D	32	90	EB	3A	AF	FC	B7	:	86
D538	D5	C9	23	7E	FE	95	C0	CD	:	5F	D708	C2	DE	D8	21	00	10	01	00	:	AA
D540	CC	D6	C0	F1	E5	21	FF	D4	:	2C	D710	30	AF	CD	56	00	21	FE	D7	:	F8
D548	22	7A	F3	CD	00	EC	E1	C9	:	F2	D718	CD	01	DC	3E	01	21	1C	D8	:	FE
D550	23	7E	FE	53	C0	23	7E	FE	:	51	D720	CD	4C	D8	3A	7C	F3	B7	3E	:	8F
D558	4D	C0	16	01	42	4A	CD	CC	:	49	D728	01	C2	F6	D7	18	07	ED	73	:	0F
D560	D6	28	28	FE	22	C0	CD	93	:	66	D730	86	EB	31	70	F3	CD	B5	DA	:	61
D568	D5	30	17	57	CD	93	D5	30	:	D8	D738	3E	02	21	26	D8	CD	4C	D8	:	50
D570	11	47	CD	93	D5	30	0B	4F	:	17	D740	CD	F8	DB	21	30	D8	CD	01	:	97
D578	23	7E	B7	28	0E	FE	22	C0	:	6E	D748	DC	11	93	EB	2A	BC	EB	CD	:	09
D580	18	05	7E	FE	22	20	04	CD	:	AC	D750	F3	E1	AF	12	21	93	EB	CD	:	01
D588	CC	D6	C0	F1	7A	E5	CD	D8	:	57	D758	01	DC	21	3F	D8	CD	01	DC	:	BF
D590	D6	E1	C9	23	7E	B7	C8	FE	:	9E	D760	11	93	EB	2A	A9	EB	CD	BA	:	D4
D598	3A	C8	FE	22	C8	CD	AC	DD	:	40	D768	E1	AF	12	21	93	EB	CD	01	:	0F
D5A0	1E	00	FE	58	28	0C	1C	FE	:	C2	D770	DC	21	42	D8	CD	01	DC	CD	:	8E
D5A8	43	28	07	1C	FE	50	28	02	:	06	D778	F8	DB	3A	90	EB	B7	28	0C	:	73
D5B0	F1	C9	7B	37	C9	CD	CC	D6	:	A4	D780	3C	3C	28	08	3E	0C	CD	A5	:	64
D5B8	FE	22	C0	23	22	84	EB	06	:	9A	D788	00	DA	96	D8	CD	F8	DB	3A	:	22
D5C0	00	7E	B7	28	0F	FE	22	28	:	B4	D790	7E	F3	B7	28	5F	21	00	00	:	D0
D5C8	07	23	04	CB	78	28	F2	C9	:	54	D798	ED	5B	A9	EB	E5	B7	ED	52	:	B7
D5D0	CD	CC	D6	C0	78	32	83	EB	:	47	D7A0	E1	30	3E	E5	CD	84	DB	2A	:	8A
D5D8	F1	E5	2A	76	F6	22	8E	EB	:	07	D7A8	9F	EB	11	93	EB	CD	F3	E1	:	BA
D5E0	21	5D	D6	CD	BD	D6	0E	00	:	C2	D7B0	3E	20	EB	77	23	77	21	9F	:	1A
D5E8	CD	B7	00	38	6B	C5	21	54	:	61	D7B8	EB	06	04	77	23	10	FC	36	:	D1
D5F0	D6	E5	ED	73	86	EB	CD	1C	:	75	D7C0	00	21	93	EB	3A	7E	F3	3D	:	87
D5F8	D9	C1	C1	38	58	C5	22	81	:	53	D7C8	20	08	CD	F8	DB	CD	01	DC	:	72
D600	EB	06	00	7E	B7	28	04	23	:	75	D7D0	18	0B	D1	D5	7B	E6	03	CC	:	F9
D608	04	18	F8	78	32	80	EB	D5	:	FE	D7D8	E3	DB	CD	EC	DB	E1	23	18	:	6E
D610	CD	80	D6	D1	C1	B7	28	D0	:	64	D7E0	B7	3A	7E	F3	3D	20	05	CD	:	91
D618	79	E6	03	CC	B4	D6	C5	EB	:	68	D7E8	F8	DB	18	08	CD	E3	DB	3E	:	BC
D620	11	93	EB	CD	BA	E1	AF	12	:	B8	D7F0	0C	CD	A5	00	3E	02	32	79	:	69
D628	21	93	EB	CD	BD	D6	21	C6	:	E6	D7F8	F3	ED	7B	86	EB	C9	0D	0A	:	AC
D630	D6	CD	BD	D6	C1	0C	CD	9C	:	6C	D800	4D	53	58	20	53	65	6C	66	:	A2
D638	00	28	AD	CD	9F	00	FE	20	:	5F	D808	20	41	73	73	65	6D	62	6C	:	E7
D640	20	A6	CD	56	01	CD	B7	00	:	6E	D810	65	72	20	20	52	65	76	20	:	64
D648	38	0E	CD	9C	00	28	F6	CD	:	9A	D818	31	2E	30	00	0D	0A	0A	50	:	00
D650	56	01	18	94	E1	CD	B4	D6	:	3B	D820	61	73	73	20	31	00	0D	0A	:	AF
D658	CD	B4	D6	E1	C9	0D	0A	41	:	59	D828	0A	50	61	73	73	20	32	00	:	F3
D660	73	73	65	6D	62	6C	65	72	:	5D	D830	0D	0A	45	6E	64	20	41	64	:	F3
D668	20	73	6F	75	72	63	65	20	:	D1	D838	64	72	65	73	73	20	00	20	:	61
D670	73	74	72	69	6E	67	20	73	:	2A	D840	2C	00	20	4C	61	62	65	6C	:	2C
D678	65	61	72	63	68	0D	0A	00	:	1A	D848	28	73	29	00	32	AD	EB	CD	:	5B
D680	3A	83	EB	B7	C8	2A	84	EB	:	C0	D850	01	DC	2A	76	F6	22	8E	EB	:	0E
D688	4F	3A	80	EB	B7	C8	ED	5B	:	BB	D858	CD	2C	DC	CD	F8	DB	2A	E6	:	85
D690	81	EB	91	47	3E	00	D8	04	:	5E	D860	EB	7C	B5	F5	28	12	E5	CD	:	FD
D698	3C	F5	C5	D5	E5	1A	BE	20	:	A8	D868	F8	DB	E1	11	93	EB	CD	BA	:	CA
D6A0	0A	13	23	0D	20	F7	E1	D1	:	16	D870	E1	AF	12	21	93	EB	18	03	:	5C
D6A8	C1	F1	C9	E1	D1	C1	F1	13	:	F2	D878	21	89	D8	CD	01	DC	21	8C	:	D9
D6B0	10	E6	AF	C9	E5	21	C9	D6	:	13	D880	D8	CD	01	DC	F1	C8	C3	06	:	04
D6B8	CD	BD	D6	E1	C9	7E	B7	C8	:	07	D888	D9	4E	6F	00	20	45	72	72	:	DF
D6C0	CD	A2	00	23	18	F7	20	20	:	E1	D890	6F	72	28	73	29	00	21	9C	:	62
D6C8	00	0D	0A	00	23	7E	B7	C8	:	37	D898	D8	C3	03	D9	0D	0A	25	20	:	D3

D8A0	41	62	6F	72	74	65	64	00	:	C1	DAB0	DB	C3	E3	DB	20	20	20	20	:	DC
D8A8	21	AE	D8	C3	03	D9	0D	0A	:	5D	DAB8	00	20	20	20	20	50	41	47	:	58
D8B0	25	20	4F	62	6A	65	63	74	:	9C	DA90	45	00	E5	CD	3C	DB	E1	30	:	1F
D8B8	20	61	72	65	61	20	66	75	:	B4	DA98	19	11	06	00	19	ED	5B	9F	:	30
D8C0	6C	6C	00	21	C9	D8	C3	03	:	60	DAA0	EB	73	23	72	AF	C9	E5	CD	:	1D
D8C8	D9	0D	0A	25	20	4C	61	62	:	44	DAA8	3C	DB	E1	38	05	CD	5E	DB	:	3B
D8D0	65	6C	20	74	61	62	6C	65	:	F9	DAB0	AF	C9	F6	FF	C9	2A	A9	EB	:	F4
D8D8	20	66	75	6C	6C	00	21	E4	:	D8	DAB8	7C	B5	C8	2B	7C	B5	C8	01	:	1E
D8E0	D8	C3	03	D9	0D	0A	25	20	:	D3	DAC0	00	00	50	59	C5	D5	C5	EB	:	F3
D8E8	53	63	72	65	65	6E	20	6E	:	EE	DAC8	11	A1	EB	CD	87	DB	C1	D1	:	5E
D8F0	6F	74	20	34	30	58	32	34	:	25	DAD0	E1	23	C5	D5	E5	CD	84	DB	:	AF
D8F8	20	74	65	78	74	20	6D	6F	:	E1	DAD8	11	A1	EB	21	99	EB	CD	75	:	84
D900	64	65	00	CD	01	DC	CD	F8	:	38	DAE0	DB	E1	D1	28	13	38	11	54	:	65
D908	DB	3E	FF	C3	F6	D7	3A	7C	:	5E	DAE8	5D	D5	E5	01	08	00	11	A1	:	D2
D910	F3	B7	28	08	ED	73	AB	EB	:	D0	DAF0	EB	21	99	EB	ED	B0	E1	D1	:	DF
D918	AF	C3	F6	D7	CD	B7	00	DA	:	9D	DAF8	23	C1	D5	E5	EB	2A	A9	EB	:	47
D920	96	D8	2A	8E	EB	5E	23	56	:	E8	DB00	2B	B7	ED	52	E1	D1	30	CA	:	CD
D928	23	ED	53	8E	EB	7A	B3	28	:	31	DB08	D5	EB	B7	ED	42	D1	28	1F	:	BE
D930	24	5E	23	56	23	D5	11	57	:	5B	DB10	C5	D5	C5	EB	11	A1	EB	CD	:	B4
D938	D9	01	20	03	7E	B7	CA	5A	:	56	DB18	87	DB	E1	E5	CD	84	DB	E1	:	35
D940	D9	B9	28	F8	0E	00	1A	BE	:	98	DB20	E3	11	99	EB	CD	94	DB	E1	:	95
D948	C2	5A	D9	13	23	10	ED	D1	:	F9	DB28	11	A1	EB	CD	94	DB	C1	03	:	9D
D950	22	8C	EB	B7	C9	37	C9	3A	:	53	DB30	2A	A9	EB	2B	2B	B7	ED	42	:	FA
D958	8F	E6	CD	F8	DB	E1	11	93	:	9A	DB38	D2	C2	DA	C9	EB	01	00	00	:	23
D960	EB	CD	BA	E1	AF	12	CD	01	:	E2	DB40	2A	A9	EB	B7	ED	42	C8	60	:	CC
D968	DC	21	6F	D9	18	95	3A	20	:	4C	DB48	69	C5	D5	CD	84	DB	D1	21	:	21
D970	20	20	20	25	20	41	73	73	:	CC	DB50	99	EB	CD	75	DB	28	04	C1	:	8E
D978	65	6D	62	6C	65	72	20	73	:	0A	DB58	03	18	E5	C1	37	C9	E5	2A	:	D0
D980																					

DC60	CD	8C	DD	CA	57	DD	22	BA	:	10	DE40	41	4C	4C	1D	43	46	20	43	:	E2
DC68	EB	7E	CD	AC	DD	CD	A0	DD	:	09	DE48	50	20	20	0C	50	44	20	56	:	A6
DC70	DA	54	EA	3A	AD	EB	3D	28	:	4F	DE50	50	44	52	57	50	49	20	54	:	4A
DC78	09	2A	BC	EB	11	CB	EB	CD	:	6E	DE58	50	49	52	55	50	4C	20	42	:	3E
DC80	F3	E1	CD	0E	E2	CD	CF	DD	:	0A	DE60	41	41	20	41	45	43	20	0E	:	99
DC88	B7	20	38	2A	BA	EB	7E	23	:	7F	DE68	45	46	42	83	45	46	4D	84	:	AC
DC90	FE	3A	28	06	CD	40	E2	C3	:	18	DE70	45	46	53	86	45	46	57	85	:	CB
DC98	57	DD	22	BA	EB	2A	BC	EB	:	CC	DE78	49	20	20	47	4A	4E	5A	1C	:	DE
DCA0	CD	B8	E9	2A	BA	EB	CD	8C	:	96	DE80	49	20	20	48	4E	44	20	81	:	04
DCA8	DD	CA	57	DD	22	BA	EB	7E	:	20	DE88	51	55	20	82	58	20	20	04	:	E4
DCB0	CD	AC	DD	CD	A0	DD	DA	51	:	CB	DE90	58	58	20	40	41	4C	54	46	:	37
DCB8	EA	CD	0E	E2	CD	CF	DD	B7	:	D7	DE98	4D	20	20	0F	4E	20	20	20	:	4A
DCC0	CA	51	EA	F5	2A	BA	EB	7E	:	47	DEA0	4E	43	20	0D	4E	44	20	5F	:	CF
DCC8	FE	3A	CA	54	EA	B7	28	0F	:	2E	DEA8	4E	44	52	60	4E	49	20	5D	:	58
DCD0	FE	3B	28	0B	FE	20	C2	51	:	9D	DEB0	4E	49	52	5E	50	20	20	1A	:	F1
DCD8	EA	CD	8C	DD	22	BA	EB	F1	:	D8	DEB8	52	20	20	1B	44	20	20	01	:	32
DCE0	21	57	DD	E5	FE	80	38	15	:	05	DEC0	44	44	20	52	44	44	52	53	:	27
DCE8	D6	80	21	EF	DC	18	1C	9F	:	15	DEC8	44	49	20	50	44	49	52	51	:	2D
DCF0	E2	BE	E2	51	EA	C7	E2	FB	:	61	DED0	45	47	20	58	4F	50	20	45	:	08
DCF8	E2	E1	E2	2C	E3	FE	50	D2	:	D4	DED8	52	20	20	0A	52	47	20	80	:	D5
DD00	8C	E9	FE	40	D2	6F	E9	21	:	FE	DEE0	54	44	52	64	54	49	52	62	:	9F
DD08	15	DD	3D	5F	16	00	19	19	:	D6	DEE8	55	54	20	21	55	54	44	63	:	3A
DD10	7E	23	66	6F	E9	4A	E3	3D	:	C9	DEF0	55	54	49	61	4F	50	20	03	:	15
DD18	E5	40	E5	7A	E5	E7	E5	24	:	59	DEF8	55	53	48	02	45	53	20	19	:	C3
DD20	E6	6B	E6	31	E6	6E	E6	74	:	16	DF00	45	54	20	1E	45	54	49	5B	:	14
DD28	E6	71	E6	77	E6	BE	E6	CB	:	09	DF08	45	54	4E	5C	4C	20	20	12	:	E1
DD30	E6	C3	E8	2A	E7	2C	E7	2F	:	E4	DF10	4C	41	20	4A	4C	43	20	10	:	B6
DD38	E7	32	E7	35	E7	38	E7	3B	:	76	DF18	4C	43	41	49	4C	44	20	59	:	22
DD40	E7	48	E7	4E	E7	4B	E7	AB	:	28	DF20	52	20	20	13	52	41	20	4C	:	A4
DD48	E7	09	E8	05	E8	3F	E8	69	:	55	DF28	52	43	20	11	52	43	41	4B	:	E7
DD50	E8	AD	E8	EE	E8	24	E9	3A	:	9A	DF30	52	44	20	5A	53	54	20	1F	:	F6
DD58	AD	EB	3D	28	12	2A	BF	EB	:	E3	DF38	42	43	20	08	43	46	20	44	:	9A
DD60	11	C2	EB	CD	BA	E1	3E	3A	:	9E	DF40	45	54	20	18	4C	41	20	14	:	92
DD68	12	21	C1	EB	CD	05	DA	2A	:	B5	DF48	52	41	20	15	52	4C	20	16	:	9C
DD70	BE	EB	26	00	ED	5B	BC	EB	:	BE	DF50	55	42	20	07	4F	52	20	0B	:	8A
DD78	19	DA	48	EA	22	BC	EB	3A	:	28	DF58	21	5D	DF	18	14	4E	5A	5A	:	8B
DD80	AE	EB	B7	CA	3C	DC	ED	7B	:	9A	DF60	20	4E	43	43	20	50	4F	50	:	03
DD88	AF	EB	C9	23	7E	B7	C8	FE	:	81	DF68	45	50	20	4D	20	00	21	92	:	D5
DD90	3B	C8	FE	20	C0	23	18	F4	:	10	DF70	DF	3A	B1	EB	FE	03	30	15	:	F8
DD98	FE	30	38	0C	FE	3A	38	0A	:	EC	DF78	06	00	11	B2	EB	1A	BE	13	:	9F
DDA0	FE	3F	38	04	FE	5B	38	02	:	0C	DF80	23	20	04	1A	BE	28	09	23	:	73
DDA8	37	C9	B7	C9	FE	61	D8	FE	:	B5	DF88	04	7E	B7	20	ED	3E	FF	C9	:	4C
ddb0	7B	D0	E6	5F	C9	2A	BA	EB	:	28	DF90	78	C9	42	20	43	20	44	20	:	6A
ddb8	7E	FE	2C	C2	51	EA	CD	8B	:	FD	DF98	45	20	48	20	4C	20	41	20	:	9A
DDC0	DD	22	BA	EB	C9	2A	BA	EB	:	3C	DFA0	49	20	52	20	41	46	42	43	:	E7
DDC8	CD	8C	DD	C2	51	EA	C9	3A	:	36	DFA8	44	45	48	4C	53	50	49	58	:	61
DDD0	B1	EB	FE	02	38	3D	FE	05	:	14	DFB0	49	59	00	CD	B5	DD	2A	BA	:	E5
DDD8	30	39	21	B2	EB	7E	D6	41	:	BC	DFB8	EB	7E	FE	28	28	21	CD	54	:	F9
DDE0	FE	1A	30	2F	6F	26	00	11	:	1D	DFC0	E0	2A	BA	EB	7E	FE	29	CA	:	1E
DDE8	15	DE	19	5E	23	7E	93	28	:	C6	DFC8	4B	EA	3A	DE	EB	3C	20	04	:	98
DDF0	22	16	00	21	30	DE	19	19	:	99	DFD0	3E	40	18	6E	3A	DF	EB	B7	:	BF
DDF8	19	19	4F	06	03	11	B3	EB	:	39	DFD8	C2	57	EA	3A	DE	EB	C9	CD	:	9C
DE00	1A	BE	20	06	13	23	10	F8	:	3C	DFE0	8B	DD	22	BA	EB	CD	54	E0	:	30
DE08	7E	C9	0D	28	06	04	23	10	:	B9	DFE8	2A	BA	EB	7E	FE	29	C2	4B	:	81
DE10	FD	18	E8	AF	C9	00	03	04	:	7C	DFF0	EA	CD	8B	DD	CD	F2	E0	C2	:	80
DE18	0C	14	19	19	19	1A	21	23	:	C9	DFF8	57	EA	22	BA	EB	3A	DE	EB	:	0B
DE20	23	28	28	2A	31	33	33	42	:	76	E000	FE	FF	20	05	3E	30	C3	42	:	95
DE28	49	49	49	49	49	4A	4A	4A	:	4B	E008	E0	FE	0E	38	04	C6	12	18	:	18
DE30	44	43	20	06	44	44	20	05	:	5A	E010	31	3A	DF	EB	B7	C2	57	EA	:	EF
DE38	4E	44	20	09	49	54	20	17	:	8F	E018	3A	DE	EB	FE	0C	20	04	3E	:	6F

E020	10	18	1F	FE	0A	20	04	3E	:	B1	E200	F1	E6	0F	C6	30	FE	3A	38	:	4C
E028	11	18	17	FE	0B	20	04	3E	:	AB	E208	02	C6	07	12	13	C9	21	B2	:	90
E030	12	18	0F	FE	0D	20	04	3E	:	A6	E210	EB	06	06	3E	20	77	23	10	:	FF
E038	13	18	07	FE	01	C2	57	EA	:	34	E218	FC	70	23	70	2A	BA	EB	11	:	DF
E040	3E	14	32	DE	EB	C9	CD	54	:	37	E220	B2	EB	7E	CD	AC	DD	CD	98	:	D6
E048	E0	3A	DE	EB	3C	C2	57	EA	:	22	E228	DD	38	0D	23	4F	04	78	FE	:	0E
E050	2A	DA	EB	C9	AF	32	DF	EB	:	63	E230	07	30	EF	79	12	13	18	EA	:	C6
E058	67	6F	22	DA	EB	3D	32	DE	:	0A	E238	22	BA	EB	78	32	B1	EB	C9	:	D6
E060	EB	2A	BA	EB	7E	FE	2B	28	:	89	E240	2A	BA	EB	7E	FE	20	C2	54	:	81
E068	74	FE	2D	28	73	FE	27	20	:	7F	E248	EA	CD	8C	DD	CA	54	EA	11	:	39
E070	05	CD	7F	E1	18	3C	FE	30	:	B4	E250	9B	E2	06	04	7E	CD	AC	DD	:	5B
E078	38	09	FE	3A	30	05	CD	FA	:	75	E258	EB	BE	EB	C2	51	EA	13	23	:	C7
E080	E0	18	2F	CD	AC	DD	CD	A0	:	EA	E260	10	F2	CD	8C	DD	CA	51	EA	:	3D
E088	DD	38	1B	CD	0E	E2	CD	6E	:	28	E268	22	BA	EB	2A	B2	EB	E5	2A	:	9D
E090	DF	3C	28	0D	3D	32	DE	EB	:	88	E270	B4	EB	E5	2A	B6	EB	E5	CD	:	01
E098	3A	DF	EB	B7	C2	4E	EA	18	:	CD	E278	46	E0	EB	E1	22	B6	EB	E1	:	96
E0A0	24	CD	E1	E9	18	0C	FE	24	:	01	E280	22	B4	EB	E1	22	B2	EB	EB	:	4C
E0A8	C2	57	EA	23	22	BA	EB	2A	:	17	E288	E5	11	CB	EB	CD	F3	E1	3E	:	8B
E0B0	BC	EB	EB	2A	DA	EB	3A	DF	:	9A	E290	3D	32	D0	EB	E1	CD	B8	E9	:	79
E0B8	EB	B7	FA	C0	E0	19	18	02	:	6F	E298	C3	C5	DD	45	51	55	20	CD	:	3D
E0C0	ED	52	22	DA	EB	2A	BA	EB	:	F5	E2A0	46	E0	CD	C5	DD	2A	DA	EB	:	84
E0C8	CD	8C	DD	22	BA	EB	CD	EF	:	B9	E2A8	22	BC	EB	3A	AD	EB	3D	C8	:	A0
E0D0	E0	C8	FE	2B	28	07	FE	2D	:	2B	E2B0	E5	11	CB	EB	CD	F3	E1	E1	:	2E
E0D8	28	06	C3	57	EA	3E	01	01	:	72	E2B8	AF	57	5F	C3	CB	D9	CD	C5	:	5E
E0E0	3E	FF	32	DF	EB	CD	8B	DD	:	6E	E2C0	DD	3E	FF	32	AE	EB	C9	CD	:	7B
E0E8	22	BA	EB	7E	C3	6D	E0	FE	:	53	E2C8	46	E0	CD	05	EA	2A	BA	EB	:	B1
E0F0	29	C8	B7	C8	FE	3B	C8	FE	:	6F	E2D0	CD	8C	DD	C8	FE	2C	C2	57	:	41
E0F8	2C	C9	11	B2	EB	06	07	2A	:	DA	E2D8	EA	CD	8B	DD	22	BA	EB	18	:	FE
E100	BA	EB	7E	CD	AC	DD	CD	EF	:	35	E2E0	E6	CD	46	E0	CD	0A	EA	2A	:	C4
E108	E0	28	14	FE	20	28	10	FE	:	70	E2E8	BA	EB	CD	8C	DD	C8	FE	2C	:	CD
E110	2B	28	0C	FE	2D	28	08	12	:	CC	E2F0	C2	57	EA	CD	8B	DD	22	BA	:	14
E118	13	23	10	E6	C3	5D	EA	22	:	58	E2F8	EB	18	E6	2A	BA	EB	7E	FE	:	34
E120	BA	EB	EB	2B	7E	FE	44	28	:	A3	E300	27	C2	57	EA	23	7E	B7	CA	:	4C
E128	05	FE	48	28	29	23	36	00	:	F5	E308	4B	EA	FE	20	DA	57	EA	FE	:	6C
E130	21	00	00	11	B2	EB	1A	B7	:	A0	E310	7F	CA	57	EA	FE	27	20	06	:	D5
E138	C8	D6	30	FE	0A	30	6A	44	:	B4	E318	23	7E	FE	27	20	07	E5	CD	:	9F
E140	4D	29	38	65	29	38	62	09	:	DF	E320	1E	EA	E1	18	DF	CD	8C	DD	:	16
E148	38	5F	29	38	5C	4F	06	00	:	A9	E328	C8	C3	51	EA	CD	46	E0	CD	:	86
E150	09	38	56	13	18	E0	36	00	:	D8	E330	C5	DD	ED	5B	DA	EB	CB	7A	:	F4
E158	21	00	00	11	B2	EB	1A	B7	:	A0	E338	C2	5D	EA	2A	BC	EB	19	22	:	15
E160	C8	D6	30	FE	0A	38	08	D6	:	EC	E340	BC	EB	3A	AD	EB	3D	C8	C3	:	41
E168	11	FE	06	30	3C	C6	0A	4F	:	A0	E348	CB	D9	CD	B6	DF	FE	40	CA	:	0E
E170	06	00	7C	E6	F0	20	32	29	:	D3	E350	57	EA	2A	DA	EB	22	DC	EB	:	19
E178	29	29	29	09	13	18	DF	ED	:	7B	E358	F5	CD	B3	DF	C1	4F	CD	C5	:	F6
E180	5B	BA	EB	1A	FE	27	20	21	:	80	E360	DD	78	FE	09	38	05	FE	10	:	A7
E188	06	03	21	00	00	13	1A	B7	:	0E	E368	DA	8C	E4	FE	07	D2	F0	E3	:	F4
E190	CA	4B	EA	FE	20	38	12	FE	:	65	E370	FE	06	20	01	3C	87	87	87	:	F6
E198	7F	28	0E	FE	27	20	06	13	:	13	E378	47	79	FE	07	30	0B	FE	06	:	04
E1A0	1A	FE	27	20	07	65	6F	10	:	4A	E380	20	01	3C	C6	40	80	C3	1E	:	C4
E1A8	E4	C3	5D	EA	ED	53	BA	EB	:	D3	E388	EA	FE	40	20	09	3E	06	80	:	15
E1B0	C9	7D	87	9F	BC	C2	5A	EA	:	2E	E390	CD	1E	EA	C3	05	EA	FE	10	:	95
E1B8	7D	C9	D5	01	10	27	CD	E5	:	05	E398	20	06	3E	46	80	C3	1E	EA	:	F5
E1C0	E1	01	E8	03	CD	E5	E1	01	:	61	E3A0	FE	20	28	07	FE	21	20	10	:	9C
E1C8	64	00	CD	E5	E1	01	0A	00	:	02	E3A8	3E	FD	21	3E	DD	C5	CD	1E	:	27
E1D0	CD	E5	E1	7D	F6	30	12	13	:	5B	E3B0	EA	C1	CD	9A	E3	C3	12	EA	:	B4
E1D8	E1	06	04	7E	FE	30	C0	36	:	8D	E3B8	78	FE	38	C2	57	EA	79	FE	:	28
E1E0	20	23	10	F7	C9	3E	30	B7	:	38	E3C0	07	20	05	3E	57	C3	64	E4	:	CC
E1E8	ED	42	38	03	3C	18	F8	09	:	BF	E3C8	FE	08	20	05	3E	5F	C3	64	:	EF
E1F0	12	13	C9	7C	CD	F8	E1	7D	:	8D	E3D0	E4	FE	11	20	05	3E	0A	C3	:	23
E1F8	F5	0F	0F	0F	0F	CD	01	E2	:	E1	E3D8	1E	EA	FE	12	20	05	3E	1A	:	95


```

E3E0 C3 1E EA FE 30 C2 57 EA : FC
E3E8 3E 3A CD 1E EA C3 0A EA : 04
E3F0 79 FE 07 30 2C FE 06 20 : FE
E3F8 01 3C 4F 78 FE 10 20 06 : 38
E400 3E 70 81 C3 1E EA FE 20 : 18
E408 28 07 FE 21 20 41 3E FD : EA
E410 21 3E DD C5 CD 1E EA C1 : 97
E418 CD 00 E4 2A DC EB C3 15 : 7A
E420 EA FE 40 C2 ED E4 78 FE : 31
E428 10 20 08 3E 36 CD 1E EA : 81
E430 C3 05 EA FE 20 28 08 FE : FE
E438 21 C2 57 EA 3E FD 21 3E : 8E
E440 DD CD 1E EA 3E 36 CD 1E : 11
E448 EA CD 18 E4 C3 05 EA 79 : E1
E450 FE 07 C2 57 EA 78 FE 07 : 85
E458 20 04 3E 47 18 06 FE 08 : CD
E460 20 0C 3E 4F F5 3E ED CD : A6
E468 1E EA F1 C3 1E EA FE 11 : D3
E470 20 05 3E 02 C3 1E EA FE : 2E
E478 12 20 05 3E 12 C3 1E EA : 52
E480 FE 30 C2 57 EA 3E 32 CD : 6E
E488 1E EA 18 71 79 FE 40 20 : 68
E490 1F 78 D6 0A FE 04 30 07 : B0
E498 87 87 87 87 3C 18 0B 3E : B9
E4A0 DD 28 02 3E FD CD 1E EA : 17
E4A8 3E 21 CD 1E EA C3 0A EA : EB
E4B0 FE 30 20 18 78 FE 0C 20 : 08
E4B8 08 3E 2A CD 1E EA C3 0A : 12
E4C0 EA D6 0A FE 06 D2 57 EA : E1
E4C8 16 08 18 49 78 FE 0D C2 : C4
E4D0 57 EA 79 FE 0C 28 11 FE : FB
E4D8 0E 28 08 FE 0F C2 57 EA : 4E
E4E0 3E FD 01 3E DD CD 1E EA : 2C
E4E8 3E F9 C3 1E EA 78 FE 30 : A8
E4F0 C2 57 EA 79 FE 0C 20 0E : B4
E4F8 3E 22 CD 1E EA 2A DC EB : 26
E500 22 DA EB C3 0A EA D6 0A : 7E
E508 FE 06 D2 57 EA 16 00 2A : 57
E510 DC EB 22 DA EB FE 04 30 : E0
E518 10 87 87 87 87 82 C6 43 : 87
E520 F5 3E ED CD 1E EA F1 18 : FE
E528 0E 3E DD 28 02 3E FD D5 : 63
E530 CD 1E EA D1 3E 22 82 CD : 55
E538 1E EA C3 0A EA 3E 04 FE : FF
E540 AF F5 CD B6 DF CD C5 DD : 75
E548 D1 3A DE EB FE 09 28 14 : 17
E550 FE 10 D6 0A FE 03 CA 57 : 10
E558 EA FE 04 30 0D 87 87 87 : BE
E560 87 C6 C1 01 3E F1 82 C3 : 83
E568 1E EA 3E DD 28 02 3E FD : 88
E570 D5 CD 1E EA F1 C6 E1 C3 : 05
E578 1E EA CD B6 DF CD B5 DD : C9
E580 3A DE EB FE 09 20 1A 16 : 5A
E588 41 CD DD E5 16 46 CD DD : D6
E590 E5 16 27 CD DD E5 CD 8C : 0A
E598 DD C2 57 EA 3E 08 C3 1E : 07
E5A0 EA F5 CD B6 DF CD C5 DD : B0
E5A8 C1 3A DE EB 4F 78 FE 0B : 94
E5B0 20 0B 79 FE 0C C2 57 EA : B1
E5B8 3E EB C3 1E EA FE 13 C2 : C7

```

```

E5C0 57 EA 79 FE 0C 28 11 FE : FB
E5C8 0E 28 08 FE 0F C2 57 EA : 4E
E5D0 3E FD 01 3E DD CD 1E EA : 2C
E5D8 3E E3 C3 1E EA 7E CD AC : E3
E5E0 DD BA C2 57 EA 23 C9 CD : 53
E5E8 5B E6 20 05 06 80 C3 82 : 31
E5F0 E6 FE 0C 28 1E FE 0E 28 : 6A
E5F8 08 FE 0F C2 57 EA 3E FD : 53
E600 21 3E DD C5 CD 1E EA C1 : 97
E608 79 FE 0C CA 57 EA B8 20 : 66
E610 02 0E 0C 79 D6 0A FE 04 : 77
E618 D2 57 EA 87 87 87 87 C6 : F5
E620 09 C3 1E EA CD 5B E6 20 : 02
E628 04 06 88 18 55 06 08 18 : 25
E630 0C CD 5B E6 20 05 06 98 : DD
E638 C3 82 E6 06 00 FE 0C C2 : FD
E640 57 EA 79 D6 0A FE 04 D2 : 6E
E648 57 EA 87 87 87 87 80 C6 : A3
E650 42 F5 3E ED CD 1E EA F1 : 28
E658 C3 1E EA CD B6 DF F5 CD : EF
E660 B3 DF C1 4F CD C5 DD 78 : 89
E668 FE 06 C9 3E 90 01 3E A0 : 7A
E670 01 3E A8 01 3E B0 01 3E : 15
E678 B8 F5 CD B6 DF C1 4F CD : EC
E680 C5 DD 79 FE 07 30 07 FE : 55
E688 06 20 01 3C 18 2C FE 10 : B5
E690 28 26 FE 20 28 12 FE 21 : C5
E698 28 11 FE 40 C2 57 EA 3E : B8
E6A0 46 80 CD 1E EA C3 05 EA : 4D
E6A8 3E DD 21 3E FD C5 CD 1E : 27
E6B0 EA C1 CD B8 E6 C3 12 EA : D5
E6B8 3E 06 80 C3 1E EA CD B6 : 12
E6C0 DF F5 CD C5 DD F1 01 03 : 38
E6C8 04 18 0B CD B6 DF F5 CD : 4B
E6D0 C5 DD F1 01 0B 05 FE 07 : A9
E6D8 30 0C FE 06 20 01 3C 87 : 24
E6E0 87 87 80 C3 1E EA FE 10 : 67
E6E8 28 18 FE 20 28 07 FE 21 : AC
E6F0 20 16 3E FD 21 3E DD C5 : 72
E6F8 CD 1E EA C1 CD 02 E7 C3 : 0F
E700 12 EA 3E 30 80 C3 1E EA : B5
E708 D6 0A FE 06 D2 57 EA FE : F5
E710 04 30 06 87 87 87 87 18 : 6E
E718 0D 3E DD 28 02 3E FD C5 : 52
E720 CD 1E EA C1 3E 20 81 C3 : 38
E728 1E EA AF 01 3E 01 01 3E : 36
E730 02 01 3E 03 01 3E 04 01 : 88
E738 3E 05 01 3E 07 F5 CD B6 : 01
E740 DF C1 4F CD C5 DD 18 24 : 9A
E748 3E 08 01 3E 10 01 3E 18 : EC
E750 F5 CD B6 DF C1 FE 40 C2 : 18
E758 57 EA 2A DA EB 7D E6 F8 : 8B
E760 B4 C2 57 EA 78 85 F5 CD : 76
E768 B3 DF C1 4F 79 FE 20 28 : 61
E770 07 FE 21 20 16 3E FD 21 : B8
E778 3E DD C5 CD 1E EA 3E CB : BE
E780 CD 1E EA CD 12 EA C1 0E : 6D
E788 06 18 18 0E 06 FE 10 28 : 80
E790 0B FE 07 D2 57 EA FE 06 : 27
E798 20 01 3C 4F C5 3E CB CD : 47

```


E7A0	1E	EA	C1	78	87	87	87	81	:	57	E980	27	2F	3F	37	00	76	F3	FB	:	30
E7A8	C3	1E	EA	2A	BA	EB	7E	FE	:	16	E988	07	17	0F	1F	D6	50	5F	16	:	E7
E7B0	28	28	27	E5	CD	87	E8	38	:	D0	E990	00	21	A3	E9	19	7E	F5	3E	:	77
E7B8	0D	3C	CA	57	EA	3D	87	87	:	9F	E998	ED	CD	1E	EA	F1	CD	1E	EA	:	88
E7C0	87	C6	C2	E1	18	06	E1	22	:	11	E9A0	C3	C5	DD	A0	B0	A8	B8	A1	:	B6
E7C8	BA	EB	3E	C3	F5	CD	46	E0	:	8E	E9A8	B1	A9	B9	44	6F	67	4D	45	:	BF
E7D0	CD	C5	DD	F1	CD	1E	EA	C3	:	F8	E9B0	A2	B2	AA	BA	A3	B3	AB	BB	:	74
E7D8	0A	EA	CD	B6	DF	F5	CD	C5	:	DD	E9B8	22	B8	EB	3A	AD	EB	3D	20	:	F4
E7E0	DD	F1	FE	10	28	1A	FE	20	:	3C	E9C0	0C	21	B2	EB	CD	A6	DA	B7	:	CE
E7E8	28	08	FE	21	C2	57	EA	06	:	58	E9C8	C8	3E	0A	18	11	E5	21	B2	:	F1
E7F0	FD	21	06	DD	2A	DA	EB	7C	:	6C	E9D0	EB	CD	92	DA	2A	B8	EB	D1	:	C2
E7F8	B5	C2	57	EA	78	CD	1E	EA	:	05	E9D8	B7	ED	52	C8	3E	0E	C3	21	:	EE
E800	3E	E9	C3	1E	EA	3E	10	18	:	58	E9E0	EB	21	B2	EB	3A	AD	EB	3D	:	B8
E808	1C	2A	BA	EB	E5	CD	87	E8	:	0C	E9E8	20	0A	CD	92	DA	2A	B8	EB	:	30
E810	38	0D	FE	04	D2	57	EA	87	:	E1	E9F0	B7	C8	18	0D	CD	92	DA	2A	:	07
E818	87	87	C6	20	E1	18	06	E1	:	D4	E9F8	B8	EB	B7	C8	3E	12	CD	21	:	60
E820	22	BA	EB	3E	18	F5	CD	46	:	25	EA00	EB	21	00	00	C9	3A	DA	EB	:	D4
E828	E0	CD	C5	DD	F1	CD	1E	EA	:	15	EA08	18	14	CD	05	EA	3A	DB	EB	:	E8
E830	2A	DA	EB	ED	5B	BC	EB	B7	:	95	EA10	18	0C	2A	DA	EB	3A	AD	EB	:	E5
E838	ED	52	2B	2B	C3	15	EA	2A	:	81	EA18	3D	28	03	CD	B1	E1	47	3A	:	48
E840	BA	EB	E5	CD	87	E8	38	0D	:	0B	EA20	AD	EB	3D	28	18	3A	BE	EB	:	F8
E848	3C	CA	57	EA	3D	87	87	87	:	19	EA28	FE	04	30	0D	87	5F	16	00	:	3B
E850	C6	C4	E1	18	06	E1	22	BA	:	46	EA30	21	D0	EB	19	EB	78	CD	F8	:	1D
E858	EB	3E	CD	F5	CD	46	E0	CD	:	AB	EA38	E1	78	CD	AC	D9	21	BE	EB	:	75
E860	C5	DD	F1	CD	1E	EA	C3	0A	:	35	EA40	34	7E	FE	80	D2	51	EA	C9	:	06
E868	EA	2A	BA	EB	CD	8C	DD	3E	:	2D	EA48	3E	00	01	3E	02	01	3E	04	:	C2
E870	C9	CA	1E	EA	CD	87	E8	D2	:	A9	EA50	01	3E	06	01	3E	08	01	3E	:	CB
E878	51	EA	3C	CA	57	EA	3D	87	:	46	EA58	0C	01	3E	10	01	3E	14	CD	:	7B
E880	87	87	C6	C0	C3	1E	EA	CD	:	2C	EA60	21	EB	C3	57	DD	7B	EA	8C	:	F4
E888	0E	E2	CD	58	DF	47	2A	BA	:	1F	EA68	EA	9A	EA	AB	EA	B8	EA	C4	:	69
E890	EB	CD	8C	DD	22	BA	EB	28	:	10	EA70	EA	DB	EA	E9	EA	F5	EA	05	:	66
E898	11	FE	2C	C2	51	EA	CD	8B	:	90	EA78	EB	15	EB	41	64	64	72	65	:	CB
E8A0	DD	22	BA	EB	CA	57	EA	78	:	27	EA80	73	73	20	4F	76	65	72	66	:	08
E8A8	B7	C9	78	37	C9	CD	46	E0	:	EB	EA88	6C	6F	77	00	42	61	6C	61	:	C2
E8B0	CD	C5	DD	2A	DA	EB	7D	E6	:	C1	EA90	6E	63	65	20	45	72	72	6F	:	EE
E8B8	C7	B4	C2	57	EA	3E	C7	B5	:	38	EA98	72	00	45	78	70	72	65	73	:	E9
E8C0	C3	1E	EA	CD	46	E0	CD	C5	:	50	EAA0	73	69	6F	6E	20	45	72	72	:	02
E8C8	DD	2A	DA	EB	7C	B7	C2	57	:	18	EAA8	6F	72	00	46	6F	72	6D	61	:	D6
E8D0	EA	7D	FE	03	D2	57	EA	F5	:	70	EAB0	74	20	45	72	72	6F	72	00	:	9E
E8D8	3E	ED	CD	1E	EA	F1	06	46	:	3D	EAB8	4C	61	62	65	6C	20	45	72	:	B7
E8E0	B7	28	07	06	56	3D	28	02	:	A9	EAC0	72	6F	72	00	4D	75	6C	74	:	F5
E8E8	06	5E	78	C3	1E	EA	CD	5B	:	CF	EAC8	69	70	6C	79	20	44	65	66	:	ED
E8F0	E6	20	17	04	79	FE	30	20	:	E8	EAD0	69	6E	65	64	20	4C	61	62	:	CF
E8F8	17	3E	DB	CD	1E	EA	2A	DA	:	09	EAD8	65	6C	00	4F	70	65	72	61	:	C8
E900	EB	7C	B7	C2	57	EA	7D	C3	:	61	EAE0	6E	64	20	45	72	72	6F	72	:	FC
E908	1E	EA	FE	06	D2	57	EA	79	:	98	EAE8	00	50	68	61	73	65	20	45	:	56
E910	FE	14	C2	57	EA	C5	3E	ED	:	05	EAFO	72	72	6F	72	00	52	65	66	:	E2
E918	CD	1E	EA	F1	87	87	87	C6	:	21	EAFO	65	72	65	6E	63	65	20	45	:	D7
E920	40	C3	1E	EA	CD	B6	DF	2A	:	97	EB00	72	72	6F	72	00	55	6E	64	:	EC
E928	DA	EB	E5	F5	CD	B3	DF	C1	:	BF	EB08	65	66	69	6E	65	64	20	4C	:	D7
E930	4F	CD	C5	DD	E1	22	DA	EB	:	86	EB10	61	62	65	6C	00	56	61	6C	:	B7
E938	79	FE	06	20	17	0C	78	FE	:	36	EB18	75	65	20	45	72	72	6F	72	:	04
E940	30	20	17	3E	D3	CD	1E	EA	:	4D	EB20	00	6F	26	00	11	65	EA	19	:	0E
E948	2A	DA	EB	7C	B7	C2	57	EA	:	25	EB28	5E	23	56	1A	D5	32	C9	EB	:	AC
E950	7D	C3	1E	EA	FE	06	D2	57	:	75	EB30	21	5C	EB	3E	FF	CD	2E	DA	:	7A
E958	EA	78	FE	14	C2	57	EA	C5	:	3C	EB38	2A	BF	EB	11	E0	EB	CD	BA	:	37
E960	3E	ED	CD	1E	EA	C1	79	87	:	C1	EB40	E1	AF	12	21	E0	EB	CD	2E	:	89
E968	87	87	C6	41	C3	1E	EA	D6	:	B6	EB48	DA	21	60	EB	AF	CD	2E	DA	:	CA
E970	40	5F	16	00	21	7F	E9	19	:	57	EB50	2A	E6	EB	23	22	E6	EB	E1	:	F2
E978	7E	CD	1E	EA	C3	C5	DD	D9	:	91	EB58	AF	C3	2E	DA	0D	0A	20	00	:	B1

EB60 3A 20 20 00 00 00 00 00 : 7A	EBD8 20 00 00 00 00 00 06 00 : 26
EB68 00 00 00 00 00 00 00 00 : 00	EBE0 00 00 00 00 00 00 00 00 : 00
EB70 00 00 00 00 00 00 00 00 : 00	EBE8 00 00 00 00 00 00 00 00 : 00
EB78 00 00 00 00 00 00 00 00 : 00	EBF0 00 00 00 00 00 00 00 00 : 00
EB80 0A 46 C0 01 23 F4 EB D1 : E4	EBF8 00 00 00 00 00 00 00 00 : 00
EB88 00 00 0B 00 46 C0 00 00 : 11	
EB90 FE 00 00 44 33 30 36 20 : FB	
EB98 20 43 43 20 20 20 20 20 : 46	
EBA0 20 20 20 00 00 00 00 00 : 60	
EBA8 00 01 00 00 00 02 00 6C : 6F	
EBB0 F3 01 41 20 20 20 20 20 : D5	
EBB8 00 00 50 C0 07 D3 00 32 : 1C	
EBC0 00 20 20 20 20 20 20 20 : E0	
EBC8 20 20 20 20 20 20 20 20 : 00	
EBD0 20 20 20 20 20 20 20 20 : 00	

リスト 4-8 Z80Aの全ニモニク、擬似命令のアセンブラ・リスト

```

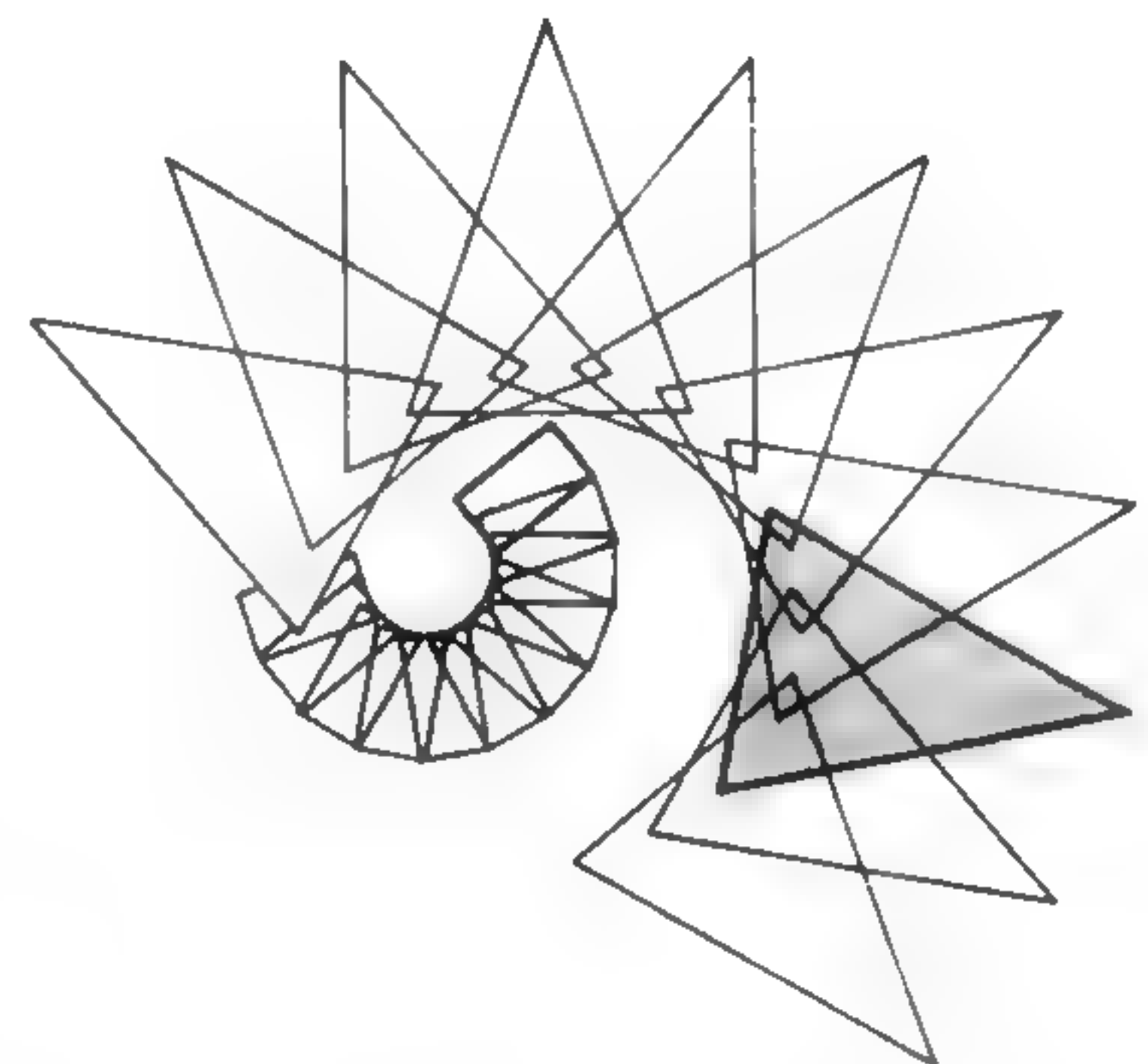
1000: ;
1010: ; Z80 All Mnemonic No.1
1020: ; OP Code ( 00-FF )
1030: ;
1040: 0012 = N EQU 12H
1050: 1234 = NN EQU 1234H
1060: 0080 = IO EQU 80H
1070:
1080: 0000 00 NOP
1090: 0001 013412 LD BC,NN
1100: 0004 02 LD (BC),A
1110: 0005 03 INC BC
1120: 0006 04 INC B
1130: 0007 05 DEC B
1140: 0008 0612 LD B,N
1150: 000A 07 RLCA
1160: 000B 08 EX AF,AF
1170: 000C 09 ADD HL,BC
1180: 000D 0A LD A,(BC)
1190: 000E 0B DEC BC
1200: 000F 0C INC C
1210: 0010 0D DEC C
1220: 0011 0E12 LD C,N
1230: 0013 0F RRCA
1240:
1250: 0014 1014 DJNZ AD1
1260: 0016 113412 LD DE,NN
1270: 0019 12 LD (DE),A
1280: 001A 13 INC DE
1290: 001B 14 INC D
1300: 001C 15 DEC D
1310: 001D 1612 LD D,N
1320: 001F 17 RLA

```


1330:	0020	1808	JR AD1
1340:	0022	19	ADD HL,DE
1350:	0023	1A	LD A,(DE)
1360:	0024	1B	DEC DE
1370:	0025	1C	INC E
1380:	0026	1D	DEC E
1390:	0027	1E12	LD E,N
1400:	0029	1F	RRA
1410:	002A		AD1:
1420:	002A	20FE	JR NZ,AD1
1430:	002C	213412	LD HL,NN
1440:	002F	223412	LD (NN),HL
1450:	0032	23	INC HL
1460:	0033	24	INC H
1470:	0034	25	DEC H
1480:	0035	2612	LD H,N
1490:	0037	27	DAA
1500:	0038	28F0	JR Z,AD1
1510:	003A	29	ADD HL,HL
1520:	003B	2A3412	LD HL,(NN)
1530:	003E	2B	DEC HL
1540:	003F	2C	INC L
1550:	0040	2D	DEC L
1560:	0041	2E12	LD L,N
1570:	0043	2F	CPL
1580:			
1590:	0044	30E4	JR NC,AD1
1600:	0046	313412	LD SP,NN
1610:	0049	323412	LD (NN),A
1620:	004C	33	INC SP
1630:	004D	34	INC (HL)
1640:	004E	35	DEC (HL)
1650:	004F	3612	LD (HL),N
1660:	0051	37	SCF
1670:	0052	38D6	JR C,AD1
1680:	0054	39	ADD HL,SP
1690:	0055	3A3412	LD A,(NN)
1700:	0058	3B	DEC SP
1710:	0059	3C	INC A
1720:	005A	3D	DEC A
1730:	005B	3E12	LD A,N
1740:	005D	3F	CCF
1750:			
1760:	005E	40	LD B,B
1770:	005F	41	LD B,C
1780:	0060	42	LD B,D
1790:	0061	43	LD B,E
1800:	0062	44	LD B,H

1810:	0063	45	LD B,L
1820:	0064	46	LD B,(HL)
1830:	0065	47	LD B,A
1840:	0066	48	LD C,B
1850:	0067	49	LD C,C
1860:	0068	4A	LD C,D
1870:	0069	4B	LD C,E
1880:	006A	4C	LD C,H
1890:	006B	4D	LD C,L
1900:	006C	4E	LD C,(HL)
1910:	006D	4F	LD C,A
1920:			
1930:	006E	50	LD D,B
1940:	006F	51	LD D,C
1950:	0070	52	LD D,D
1960:	0071	53	LD D,E
1970:	0072	54	LD D,H
1980:	0073	55	LD D,L
1990:	0074	56	LD D,(HL)
2000:	0075	57	LD D,A
2010:	0076	58	LD E,B
2020:	0077	59	LD E,C
2030:	0078	5A	LD E,D
2040:	0079	5B	LD E,E
2050:	007A	5C	LD E,H
2060:	007B	5D	LD E,L
2070:	007C	5E	LD E,(HL)
2080:	007D	5F	LD E,A
2090:			
2100:	007E	60	LD H,B
2110:	007F	61	LD H,C
2120:	0080	62	LD H,D
2130:	0081	63	LD H,E
2140:	0082	64	LD H,H
2150:	0083	65	LD H,L
2160:	0084	66	LD H,(HL)
2170:	0085	67	LD H,A
2180:	0086	68	LD L,B
2190:	0087	69	LD L,C
2200:	0088	6A	LD L,D
2210:	0089	6B	LD L,E
2220:	008A	6C	LD L,H
2230:	008B	6D	LD L,L
2240:	008C	6E	LD L,(HL)
2250:	008D	6F	LD L,A
2260:			
2270:	008E	70	LD (HL),B
2280:	008F	71	LD (HL),C

2290:	0090	72	LD (HL),D
2300:	0091	73	LD (HL),E
2310:	0092	74	LD (HL),H
2320:	0093	75	LD (HL),L
2330:	0094	76	HALT
2340:	0095	77	LD (HL),A
2350:	0096	78	LD A,B
2360:	0097	79	LD A,C
2370:	0098	7A	LD A,D
2380:	0099	7B	LD A,E
2390:	009A	7C	LD A,H
2400:	009B	7D	LD A,L
2410:	009C	7E	LD A,(HL)
2420:	009D	7F	LD A,A
2430:			
2440:	009E	80	ADD A,B
2450:	009F	81	ADD A,C
2460:	00A0	82	ADD A,D
2470:	00A1	83	ADD A,E
2480:	00A2	84	ADD A,H
2490:	00A3	85	ADD A,L
2500:	00A4	86	ADD A,(HL)
2510:	00A5	87	ADD A,A
2520:	00A6	88	ADC A,B
2530:	00A7	89	ADC A,C
2540:	00A8	8A	ADC A,D
2550:	00A9	8B	ADC A,E
2560:	00AA	8C	ADC A,H
2570:	00AB	8D	ADC A,L
2580:	00AC	8E	ADC A,(HL)
2590:	00AD	8F	ADC A,A
2600:			
2610:	00AE	90	SUB B
2620:	00AF	91	SUB C
2630:	00B0	92	SUB D
2640:	00B1	93	SUB E
2650:	00B2	94	SUB H
2660:	00B3	95	SUB L
2670:	00B4	96	SUB (HL)
2680:	00B5	97	SUB A
2690:	00B6	98	SBC A,B
2700:	00B7	99	SBC A,C
2710:	00B8	9A	SBC A,D
2720:	00B9	9B	SBC A,E
2730:	00BA	9C	SBC A,H
2740:	00BB	9D	SBC A,L
2750:	00BC	9E	SBC A,(HL)
2760:	00BD	9F	SBC A,A

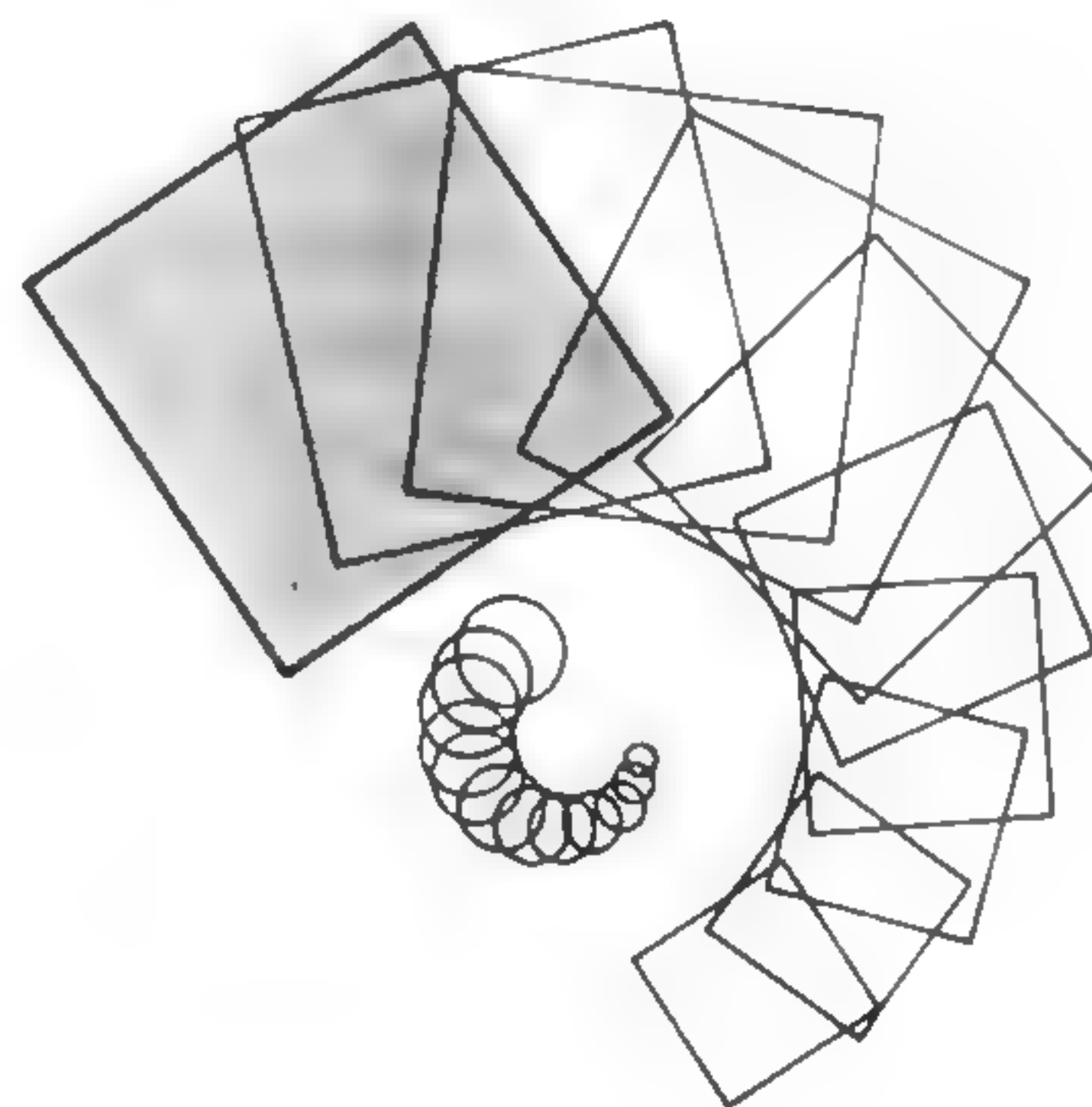


2770:			
2780:	00BE	A0	AND B
2790:	00BF	A1	AND C
2800:	00C0	A2	AND D
2810:	00C1	A3	AND E
2820:	00C2	A4	AND H
2830:	00C3	A5	AND L
2840:	00C4	A6	AND (HL)
2850:	00C5	A7	AND A
2860:	00C6	A8	XOR B
2870:	00C7	A9	XOR C
2880:	00C8	AA	XOR D
2890:	00C9	AB	XOR E
2900:	00CA	AC	XOR H
2910:	00CB	AD	XOR L
2920:	00CC	AE	XOR (HL)
2930:	00CD	AF	XOR A
2940:			
2950:	00CE	B0	OR B
2960:	00CF	B1	OR C
2970:	00D0	B2	OR D
2980:	00D1	B3	OR E
2990:	00D2	B4	OR H
3000:	00D3	B5	OR L
3010:	00D4	B6	OR (HL)
3020:	00D5	B7	OR A
3030:	00D6	B8	CF B
3040:	00D7	B9	CF C
3050:	00D8	BA	CF D
3060:	00D9	BB	CF E
3070:	00DA	BC	CF H
3080:	00DB	BD	CF L
3090:	00DC	BE	CF (HL)
3100:	00DD	BF	CF A
3110:			
3120:	00DE	C0	RET NZ
3130:	00DF	C1	POP BC
3140:	00E0	C21801	JF NZ,AD2
3150:	00E3	C31801	JF AD2
3160:	00E6	C41801	CALL NZ,AD2
3170:	00E9	C5	PUSH BC
3180:	00EA	C612	ADD A,N
3190:	00EC	C7	RST 00H
3200:	00ED	C8	RET Z
3210:	00EE	C9	RET
3220:	00EF	CA1801	JF Z,AD2
3230:	00F2	CB	DEFB 0CBH ;Rotate,Shift,Bit
3240:	00F3	CC1801	CALL Z,AD2

3250:	00F6	CD1801	CALL AD2
3260:	00F9	CE12	ADC A,N
3270:	00FB	CF	RST 08H
3280:			
3290:	00FC	D0	RET NC
3300:	00FD	D1	POP DE
3310:	00FE	D21801	JF NC,AD2
3320:	0101	D380	OUT (IO),A
3330:	0103	D41801	CALL NC,AD2
3340:	0106	D5	PUSH DE
3350:	0107	D612	SUB N
3360:	0109	D7	RST 10H
3370:	010A	D8	RET C
3380:	010B	D9	EXX
3390:	010C	DA1801	JF C,AD2
3400:	010F	DB80	IN A,(IO)
3410:	0111	DC1801	CALL C,AD2
3420:	0114	DD	DEFB 0DDH ; Index IX
3430:	0115	DE12	SBC A,N
3440:	0117	DF	RST 18H
3450:	0118		AD2:
3460:	0118	E0	RET PO
3470:	0119	E1	POP HL
3480:	011A	E21801	JF PO,AD2
3490:	011D	E3	EX (SP),HL
3500:	011E	E41801	CALL PO,AD2
3510:	0121	E5	PUSH HL
3520:	0122	E612	AND N
3530:	0124	E7	RST 20H
3540:	0125	E8	RET PE
3550:	0126	E9	JF (HL)
3560:	0127	EA1801	JF PE,AD2
3570:	012A	EB	EX DE,HL
3580:	012B	EC1801	CALL PE,AD2
3590:	012E	ED	DEFB 0EDH ; 2 Byte OP
3600:	012F	EE12	XOR N
3610:	0131	EF	RST 28H
3620:			
3630:	0132	F0	RET P
3640:	0133	F1	POP AF
3650:	0134	F21801	JF P,AD2
3660:	0137	F3	DI
3670:	0138	F41801	CALL P,AD2
3680:	013B	F5	PUSH AF
3690:	013C	F612	OR N
3700:	013E	F7	RST 30H
3710:	013F	F8	RET M
3720:	0140	F9	LD SP,HL

3730:	0141	FA1801	JF M,AD2
3740:	0144	FB	EI
3750:	0145	FC1801	CALL M,AD2
3760:	0148	FD	DEFB 0FDH ; Index IY
3770:	0149	FE12	CF N
3780:	014B	FF	RST 38H
3790:			
1000:			;
1010:			; Z80 All Mnemonic No.2
1020:			; OP Code (CB XX)
1030:			;
1040:			
1050:	0000	CB00	RLC B
1060:	0002	CB01	RLC C
1070:	0004	CB02	RLC D
1080:	0006	CB03	RLC E
1090:	0008	CB04	RLC H
1100:	000A	CB05	RLC L
1110:	000C	CB06	RLC (HL)
1120:	000E	CB07	RLC A
1130:	0010	CB08	RRC B
1140:	0012	CB09	RRC C
1150:	0014	CB0A	RRC D
1160:	0016	CB0B	RRC E
1170:	0018	CB0C	RRC H
1180:	001A	CB0D	RRC L
1190:	001C	CB0E	RRC (HL)
1200:	001E	CB0F	RRC A
1210:			
1220:	0020	CB10	RL B
1230:	0022	CB11	RL C
1240:	0024	CB12	RL D
1250:	0026	CB13	RL E
1260:	0028	CB14	RL H
1270:	002A	CB15	RL L
1280:	002C	CB16	RL (HL)
1290:	002E	CB17	RL A
1300:	0030	CB18	RR B
1310:	0032	CB19	RR C
1320:	0034	CB1A	RR D
1330:	0036	CB1B	RR E
1340:	0038	CB1C	RR H
1350:	003A	CB1D	RR L
1360:	003C	CB1E	RR (HL)
1370:	003E	CB1F	RR A
1380:			
1390:	0040	CB20	SLA B
1400:	0042	CB21	SLA C

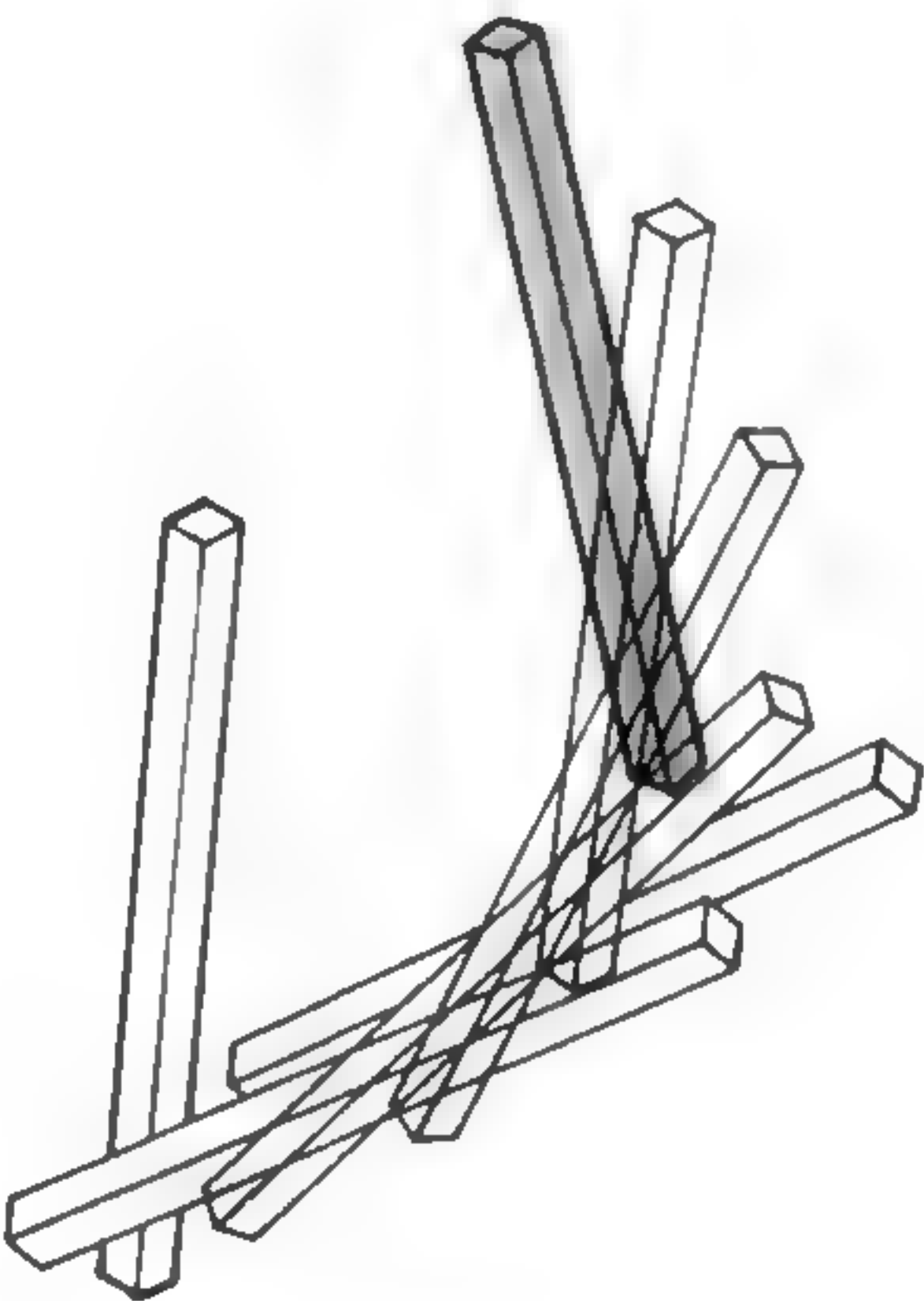
1410:	0044	CB22	SLA D
1420:	0046	CB23	SLA E
1430:	0048	CB24	SLA H
1440:	004A	CB25	SLA L
1450:	004C	CB26	SLA (HL)
1460:	004E	CB27	SLA A
1470:	0050	CB28	SRA B
1480:	0052	CB29	SRA C
1490:	0054	CB2A	SRA D
1500:	0056	CB2B	SRA E
1510:	0058	CB2C	SRA H
1520:	005A	CB2D	SRA L
1530:	005C	CB2E	SRA (HL)
1540:	005E	CB2F	SRA A
1550:			
1560:			:
1570:			:
1580:			:
1590:			:
1600:			:
1610:			:
1620:			:
1630:			:
1640:	0060	CB38	SRL B
1650:	0062	CB39	SRL C
1660:	0064	CB3A	SRL D
1670:	0066	CB3B	SRL E
1680:	0068	CB3C	SRL H
1690:	006A	CB3D	SRL L
1700:	006C	CB3E	SRL (HL)
1710:	006E	CB3F	SRL A
1720:			
1730:	0070	CB40	BIT 0,B
1740:	0072	CB41	BIT 0,C
1750:	0074	CB42	BIT 0,D
1760:	0076	CB43	BIT 0,E
1770:	0078	CB44	BIT 0,H
1780:	007A	CB45	BIT 0,L
1790:	007C	CB46	BIT 0,(HL)
1800:	007E	CB47	BIT 0,A
1810:	0080	CB48	BIT 1,B
1820:	0082	CB49	BIT 1,C
1830:	0084	CB4A	BIT 1,D
1840:	0086	CB4B	BIT 1,E
1850:	0088	CB4C	BIT 1,H
1860:	008A	CB4D	BIT 1,L
1870:	008C	CB4E	BIT 1,(HL)
1880:	008E	CB4F	BIT 1,A



1890:			
1900:	0090	CB50	BIT 2,B
1910:	0092	CB51	BIT 2,C
1920:	0094	CB52	BIT 2,D
1930:	0096	CB53	BIT 2,E
1940:	0098	CB54	BIT 2,H
1950:	009A	CB55	BIT 2,L
1960:	009C	CB56	BIT 2,(HL)
1970:	009E	CB57	BIT 2,A
1980:	00A0	CB58	BIT 3,B
1990:	00A2	CB59	BIT 3,C
2000:	00A4	CB5A	BIT 3,D
2010:	00A6	CB5B	BIT 3,E
2020:	00A8	CB5C	BIT 3,H
2030:	00AA	CB5D	BIT 3,L
2040:	00AC	CB5E	BIT 3,(HL)
2050:	00AE	CB5F	BIT 3,A
2060:			
2070:	00B0	CB60	BIT 4,B
2080:	00B2	CB61	BIT 4,C
2090:	00B4	CB62	BIT 4,D
2100:	00B6	CB63	BIT 4,E
2110:	00B8	CB64	BIT 4,H
2120:	00BA	CB65	BIT 4,L
2130:	00BC	CB66	BIT 4,(HL)
2140:	00BE	CB67	BIT 4,A
2150:	00C0	CB68	BIT 5,B
2160:	00C2	CB69	BIT 5,C
2170:	00C4	CB6A	BIT 5,D
2180:	00C6	CB6B	BIT 5,E
2190:	00C8	CB6C	BIT 5,H
2200:	00CA	CB6D	BIT 5,L
2210:	00CC	CB6E	BIT 5,(HL)
2220:	00CE	CB6F	BIT 5,A
2230:			
2240:	00D0	CB70	BIT 6,B
2250:	00D2	CB71	BIT 6,C
2260:	00D4	CB72	BIT 6,D
2270:	00D6	CB73	BIT 6,E
2280:	00D8	CB74	BIT 6,H
2290:	00DA	CB75	BIT 6,L
2300:	00DC	CB76	BIT 6,(HL)
2310:	00DE	CB77	BIT 6,A
2320:	00E0	CB78	BIT 7,B
2330:	00E2	CB79	BIT 7,C
2340:	00E4	CB7A	BIT 7,D
2350:	00E6	CB7B	BIT 7,E
2360:	00E8	CB7C	BIT 7,H

2370:	00EA	CB7D	BIT 7,L
2380:	00EC	CB7E	BIT 7,(HL)
2390:	00EE	CB7F	BIT 7,A
2400:			
2410:	00F0	CB80	RES 0,B
2420:	00F2	CB81	RES 0,C
2430:	00F4	CB82	RES 0,D
2440:	00F6	CB83	RES 0,E
2450:	00F8	CB84	RES 0,H
2460:	00FA	CB85	RES 0,L
2470:	00FC	CB86	RES 0,(HL)
2480:	00FE	CB87	RES 0,A
2490:	0100	CB88	RES 1,B
2500:	0102	CB89	RES 1,C
2510:	0104	CB8A	RES 1,D
2520:	0106	CB8B	RES 1,E
2530:	0108	CB8C	RES 1,H
2540:	010A	CB8D	RES 1,L
2550:	010C	CB8E	RES 1,(HL)
2560:	010E	CB8F	RES 1,A
2570:			
2580:	0110	CB90	RES 2,B
2590:	0112	CB91	RES 2,C
2600:	0114	CB92	RES 2,D
2610:	0116	CB93	RES 2,E
2620:	0118	CB94	RES 2,H
2630:	011A	CB95	RES 2,L
2640:	011C	CB96	RES 2,(HL)
2650:	011E	CB97	RES 2,A
2660:	0120	CB98	RES 3,B
2670:	0122	CB99	RES 3,C
2680:	0124	CB9A	RES 3,D
2690:	0126	CB9B	RES 3,E
2700:	0128	CB9C	RES 3,H
2710:	012A	CB9D	RES 3,L
2720:	012C	CB9E	RES 3,(HL)
2730:	012E	CB9F	RES 3,A
2740:			
2750:	0130	CBA0	RES 4,B
2760:	0132	CBA1	RES 4,C
2770:	0134	CBA2	RES 4,D
2780:	0136	CBA3	RES 4,E
2790:	0138	CBA4	RES 4,H
2800:	013A	CBA5	RES 4,L
2810:	013C	CBA6	RES 4,(HL)
2820:	013E	CBA7	RES 4,A
2830:	0140	CBA8	RES 5,B
2840:	0142	CBA9	RES 5,C

2850:	0144	CBAA	RES 5,D
2860:	0146	CBAB	RES 5,E
2870:	0148	CBAC	RES 5,H
2880:	014A	CBAD	RES 5,L
2890:	014C	CBAE	RES 5,(HL)
2900:	014E	CBAF	RES 5,A
2910:			
2920:	0150	CBB0	RES 6,B
2930:	0152	CBB1	RES 6,C
2940:	0154	CBB2	RES 6,D
2950:	0156	CBB3	RES 6,E
2960:	0158	CBB4	RES 6,H
2970:	015A	CBB5	RES 6,L
2980:	015C	CBB6	RES 6,(HL)
2990:	015E	CBB7	RES 6,A
3000:	0160	CBB8	RES 7,B
3010:	0162	CBB9	RES 7,C
3020:	0164	CBBA	RES 7,D
3030:	0166	CBBB	RES 7,E
3040:	0168	CBBC	RES 7,H
3050:	016A	CBBD	RES 7,L
3060:	016C	CBBE	RES 7,(HL)
3070:	016E	CBBF	RES 7,A
3080:			
3090:	0170	CBC0	SET 0,B
3100:	0172	CBC1	SET 0,C
3110:	0174	CBC2	SET 0,D
3120:	0176	CBC3	SET 0,E
3130:	0178	CBC4	SET 0,H
3140:	017A	CBC5	SET 0,L
3150:	017C	CBC6	SET 0,(HL)
3160:	017E	CBC7	SET 0,A
3170:	0180	CBC8	SET 1,B
3180:	0182	CBC9	SET 1,C
3190:	0184	CBCA	SET 1,D
3200:	0186	CBCB	SET 1,E
3210:	0188	CBCD	SET 1,H
3220:	018A	CBCD	SET 1,L
3230:	018C	CBCE	SET 1,(HL)
3240:	018E	CBCF	SET 1,A
3250:			
3260:	0190	CBD0	SET 2,B
3270:	0192	CBD1	SET 2,C
3280:	0194	CBD2	SET 2,D
3290:	0196	CBD3	SET 2,E
3300:	0198	CBD4	SET 2,H
3310:	019A	CBD5	SET 2,L
3320:	019C	CBD6	SET 2,(HL)



3330:	019E	CBD7	SET 2,A
3340:	01A0	CBD8	SET 3,B
3350:	01A2	CBD9	SET 3,C
3360:	01A4	CBDA	SET 3,D
3370:	01A6	CBDB	SET 3,E
3380:	01A8	CBDC	SET 3,H
3390:	01AA	CBDD	SET 3,L
3400:	01AC	CBDE	SET 3,(HL)
3410:	01AE	CBDF	SET 3,A
3420:			
3430:	01B0	CBE0	SET 4,B
3440:	01B2	CBE1	SET 4,C
3450:	01B4	CBE2	SET 4,D
3460:	01B6	CBE3	SET 4,E
3470:	01B8	CBE4	SET 4,H
3480:	01BA	CBE5	SET 4,L
3490:	01BC	CBE6	SET 4,(HL)
3500:	01BE	CBE7	SET 4,A
3510:	01C0	CBE8	SET 5,B
3520:	01C2	CBE9	SET 5,C
3530:	01C4	CBEA	SET 5,D
3540:	01C6	CBEB	SET 5,E
3550:	01C8	CBEC	SET 5,H
3560:	01CA	CBED	SET 5,L
3570:	01CC	CBEE	SET 5,(HL)
3580:	01CE	CBEF	SET 5,A
3590:			
3600:	01D0	CBF0	SET 6,B
3610:	01D2	CBF1	SET 6,C
3620:	01D4	CBF2	SET 6,D
3630:	01D6	CBF3	SET 6,E
3640:	01D8	CBF4	SET 6,H
3650:	01DA	CBF5	SET 6,L
3660:	01DC	CBF6	SET 6,(HL)
3670:	01DE	CBF7	SET 6,A
3680:	01E0	CBF8	SET 7,B
3690:	01E2	CBF9	SET 7,C
3700:	01E4	CBFA	SET 7,D
3710:	01E6	CBFB	SET 7,E
3720:	01E8	CBFC	SET 7,H
3730:	01EA	CBFD	SET 7,L
3740:	01EC	CBFE	SET 7,(HL)
3750:	01EE	CBFF	SET 7,A
3760:			

1000:	:	
1010:	:	Z80 All Mnemonic No.3
1020:	:	OP Code (DD XX)
1030:	:	


```

1040: 0012 = N EQU 12H
1050: 1234 = NN EQU 1234H
1060: 0056 = DSP EQU 56H
1070:
1080: 0000 DD09 ADD IX,BC
1090: 0002 DD19 ADD IX,DE
1100: 0004 DD213412 LD IX,NN
1110: 0008 DD223412 LD (NN),IX
1120: 000C DD23 INC IX
1130: 000E DD29 ADD IX,IX
1140: 0010 DD2A3412 LD IX,(NN)
1150: 0014 DD2B DEC IX
1160: 0016 DD3456 INC (IX+DSP)
1170: 0019 DD35AA DEC (IX-DSP)
1180: 001C DD365612 LD (IX+DSP),N
1190: 0020 DD39 ADD IX,SP
1200: 0022 DD4656 LD B,(IX+DSP)
1210: 0025 DD4E56 LD C,(IX+DSP)
1220: 0028 DD5656 LD D,(IX+DSP)
1230: 002B DD5E56 LD E,(IX+DSP)
1240: 002E DD6656 LD H,(IX+DSP)
1250: 0031 DD6E56 LD L,(IX+DSP)
1260: 0034 DD7056 LD (IX+DSP),B
1270: 0037 DD7156 LD (IX+DSP),C
1280: 003A DD7256 LD (IX+DSP),D
1290: 003D DD7356 LD (IX+DSP),E
1300: 0040 DD7456 LD (IX+DSP),H
1310: 0043 DD7556 LD (IX+DSP),L
1320: 0046 DD7756 LD (IX+DSP),A
1330: 0049 DD7E56 LD A,(IX+DSP)
1340: 004C DD8656 ADD A,(IX+DSP)
1350: 004F DD8E56 ADC A,(IX+DSP)
1360: 0052 DD9656 SUB (IX+DSP)
1370: 0055 DD9E56 SBC A,(IX+DSP)
1380: 0058 DDA656 AND (IX+DSP)
1390: 005B DDAE56 XOR (IX+DSP)
1400: 005E DDB656 OR (IX+DSP)
1410: 0061 DDBE56 CF (IX+DSP)
1420: 0064 DDCB5606 RLC (IX+DSP)
1430: 0068 DDCB560E RRC (IX+DSP)
1440: 006C DDCB5616 RL (IX+DSP)
1450: 0070 DDCB561E RR (IX+DSP)
1460: 0074 DDCB5626 SLA (IX+DSP)
1470: 0078 DDCB562E SRA (IX+DSP)
1480: 007C DDCB563E SRL (IX+DSP)
1490: 0080 DDCB5646 BIT 0,(IX+DSP)
1500: 0084 DDCB564E BIT 1,(IX+DSP)
1510: 0088 DDCB5656 BIT 2,(IX+DSP)

```


1520:	008C	DDCB565E	BIT 3, (IX+DSP)
1530:	0090	DDCB5666	BIT 4, (IX+DSP)
1540:	0094	DDCB566E	BIT 5, (IX+DSP)
1550:	0098	DDCB5676	BIT 6, (IX+DSP)
1560:	009C	DDCB567E	BIT 7, (IX+DSP)
1570:	00A0	DDCB5686	RES 0, (IX+DSP)
1580:	00A4	DDCB568E	RES 1, (IX+DSP)
1590:	00A8	DDCB5696	RES 2, (IX+DSP)
1600:	00AC	DDCB569E	RES 3, (IX+DSP)
1610:	00B0	DDCB56A6	RES 4, (IX+DSP)
1620:	00B4	DDCB56AE	RES 5, (IX+DSP)
1630:	00B8	DDCB56B6	RES 6, (IX+DSP)
1640:	00BC	DDCB56BE	RES 7, (IX+DSP)
1650:	00C0	DDCB56C6	SET 0, (IX+DSP)
1660:	00C4	DDCB56CE	SET 1, (IX+DSP)
1670:	00C8	DDCB56D6	SET 2, (IX+DSP)
1680:	00CC	DDCB56DE	SET 3, (IX+DSP)
1690:	00D0	DDCB56E6	SET 4, (IX+DSP)
1700:	00D4	DDCB56EE	SET 5, (IX+DSP)
1710:	00D8	DDCB56F6	SET 6, (IX+DSP)
1720:	00DC	DDCB56FE	SET 7, (IX+DSP)
1730:	00E0	DDE1	POP IX
1740:	00E2	DDE3	EX (SP), IX
1750:	00E4	DDE5	PUSH IX
1760:	00E6	DDE9	JP (IX)
1770:	00E8	DDF9	LD SP, IX
1780:			

1000:			;
1010:			; Z80 All Mnemonic No.4
1020:			; OP Code (FD XX)
1030:			;
1040:	0012 =		N EQU 12H
1050:	1234 =		NN EQU 1234H
1060:	0056 =		DSP EQU 56H
1070:			
1080:	0000	FD09	ADD IY, BC
1090:	0002	FD19	ADD IY, DE
1100:	0004	FD213412	LD IY, NN
1110:	0008	FD223412	LD (NN), IY
1120:	000C	FD23	INC IY
1130:	000E	FD29	ADD IY, IY
1140:	0010	FD2A3412	LD IY, (NN)
1150:	0014	FD2B	DEC IY
1160:	0016	FD3456	INC (IY+DSP)
1170:	0019	FD35AA	DEC (IY-DSP)
1180:	001C	FD365612	LD (IY+DSP), N
1190:	0020	FD39	ADD IY, SP
1200:	0022	FD4656	LD B, (IY+DSP)

1210:	0025	FD4E56	LD C, (IY+DSF)
1220:	0028	FD5656	LD D, (IY+DSF)
1230:	0028	FD5E56	LD E, (IY+DSF)
1240:	002E	FD6656	LD H, (IY+DSF)
1250:	0031	FD6E56	LD L, (IY+DSF)
1260:	0034	FD7056	LD (IY+DSF), B
1270:	0037	FD7156	LD (IY+DSF), C
1280:	003A	FD7256	LD (IY+DSF), D
1290:	003D	FD7356	LD (IY+DSF), E
1300:	0040	FD7456	LD (IY+DSF), H
1310:	0043	FD7556	LD (IY+DSF), L
1320:	0046	FD7756	LD (IY+DSF), A
1330:	0049	FD7E56	LD A, (IY+DSF)
1340:	004C	FD8656	ADD A, (IY+DSF)
1350:	004F	FD8E56	ADC A, (IY+DSF)
1360:	0052	FD9656	SUB (IY+DSF)
1370:	0055	FD9E56	SBC A, (IY+DSF)
1380:	0058	FDA656	AND (IY+DSF)
1390:	005B	FDAE56	XOR (IY+DSF)
1400:	005E	FDB656	OR (IY+DSF)
1410:	0061	FDBE56	CP (IY+DSF)
1420:	0064	FDCB5606	RLC (IY+DSF)
1430:	0068	FDCB560E	RRC (IY+DSF)
1440:	006C	FDCB5616	RL (IY+DSF)
1450:	0070	FDCB561E	RR (IY+DSF)
1460:	0074	FDCB5626	SLA (IY+DSF)
1470:	0078	FDCB562E	SRA (IY+DSF)
1480:	007C	FDCB563E	SRL (IY+DSF)
1490:	0080	FDCB5646	BIT 0, (IY+DSF)
1500:	0084	FDCB564E	BIT 1, (IY+DSF)
1510:	0088	FDCB5656	BIT 2, (IY+DSF)
1520:	008C	FDCB565E	BIT 3, (IY+DSF)
1530:	0090	FDCB5666	BIT 4, (IY+DSF)
1540:	0094	FDCB566E	BIT 5, (IY+DSF)
1550:	0098	FDCB5676	BIT 6, (IY+DSF)
1560:	009C	FDCB567E	BIT 7, (IY+DSF)
1570:	00A0	FDCB5686	RES 0, (IY+DSF)
1580:	00A4	FDCB568E	RES 1, (IY+DSF)
1590:	00A8	FDCB5696	RES 2, (IY+DSF)
1600:	00AC	FDCB569E	RES 3, (IY+DSF)
1610:	00B0	FDCB56A6	RES 4, (IY+DSF)
1620:	00B4	FDCB56AE	RES 5, (IY+DSF)
1630:	00B8	FDCB56B6	RES 6, (IY+DSF)
1640:	00BC	FDCB56BE	RES 7, (IY+DSF)
1650:	00C0	FDCB56C6	SET 0, (IY+DSF)
1660:	00C4	FDCB56CE	SET 1, (IY+DSF)
1670:	00C8	FDCB56D6	SET 2, (IY+DSF)
1680:	00CC	FDCB56DE	SET 3, (IY+DSF)

1690:	00D0	FDCB56E6	SET 4, (IY+DSP)
1700:	00D4	FDCB56EE	SET 5, (IY+DSP)
1710:	00D8	FDCB56F6	SET 6, (IY+DSP)
1720:	00DC	FDCB56FE	SET 7, (IY+DSP)
1730:	00E0	FDE1	POP IY
1740:	00E2	FDE3	EX (SP), IY
1750:	00E4	FDE5	PUSH IY
1760:	00E6	FDE9	JF (IY)
1770:	00E8	FDF9	LD SP, IY
1780:			

1000:			;
1010:			; Z80 All Mnemonic No.5
1020:			; OP Code (ED XX)
1030:			;
1040:	0012	=	N EQU 12H
1050:	1234	=	NN EQU 1234H
1060:			
1070:	0000	ED40	IN B, (C)
1080:	0002	ED41	OUT (C), B
1090:	0004	ED42	SBC HL, BC
1100:	0006	ED433412	LD (NN), BC
1110:	000A	ED44	NEG
1120:	000C	ED45	RETN
1130:	000E	ED46	IM 0
1140:	0010	ED47	LD I, A
1150:	0012	ED48	IN C, (C)
1160:	0014	ED49	OUT (C), C
1170:	0016	ED4A	ADC HL, BC
1180:	0018	ED4B3412	LD BC, (NN)
1190:	001C	ED4D	RETI
1200:	001E	ED4F	LD R, A
1210:	0020	ED50	IN D, (C)
1220:	0022	ED51	OUT (C), D
1230:	0024	ED52	SBC HL, DE
1240:	0026	ED533412	LD (NN), DE
1250:	002A	ED56	IM 1
1260:	002C	ED57	LD A, I
1270:	002E	ED58	IN E, (C)
1280:	0030	ED59	OUT (C), E
1290:	0032	ED5A	ADC HL, DE
1300:	0034	ED5B3412	LD DE, (NN)
1310:	0038	ED5E	IM 2
1320:	003A	ED5F	LD A, R
1330:	003C	ED60	IN H, (C)
1340:	003E	ED61	OUT (C), H
1350:	0040	ED62	SBC HL, HL
1360:	0042	ED67	RRD
1370:	0044	ED68	IN L, (C)


```

1380: 0046 ED69 OUT (C),L
1390: 0048 ED6A ADC HL,HL
1400: 004A ED6F RLD
1410: 004C ED72 SBC HL,SP
1420: 004E ED733412 LD (NN),SP
1430: 0052 ED78 IN A,(C)
1440: 0054 ED79 OUT (C),A
1450: 0056 ED7A ADC HL,SP
1460: 0058 ED7B3412 LD SP,(NN)
1470: 005C EDA0 LDI
1480: 005E EDA1 CFI
1490: 0060 EDA2 INI
1500: 0062 EDA3 OUTI
1510: 0064 EDA8 LDD
1520: 0066 EDA9 CPD
1530: 0068 EDAA IND
1540: 006A EDAB OUTD
1550: 006C EDB0 LDIR
1560: 006E EDB1 CFIR
1570: 0070 EDB2 INIR
1580: 0072 EDB3 OTIR
1590: 0074 EDB8 LDDR
1600: 0076 EDB9 CPDR
1610: 0078 ED8A INDR
1620: 007A ED8B OTDR
1630:

```

```

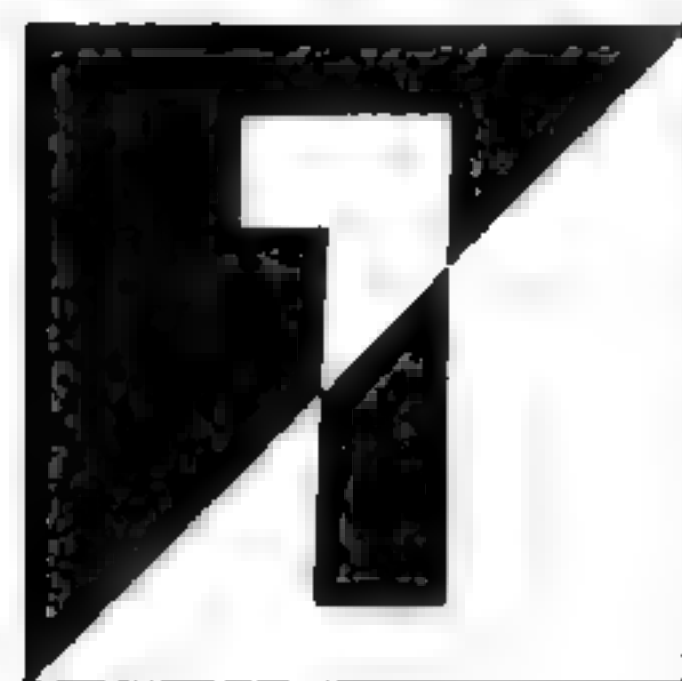
1000: ;
1010: ; Pseudo Instruction
1020: ;
1030: D400 ORG 0D400H
1040: D401 = XXX EQU $+1
1050: D400 00010203 DEFB 0,1,2,3,4,5,6,7,8
1060: D409 10F74127 DEFB 10H,0F7H,'A',' '
1070: D40D 61622763 DEFM 'ab''cd'
1080: D412 41424344 DEFM 'ABCDEFGHijkl'
1090: D41E 01000A00 DEFW 1,10
1100: D422 6400E803 DEFW 100,1000
1110: D426 DEFS 20H
1120: D446 4C4801D4 DEFW 'HL',XXX
1130: D44A END

```



5章 マシン語プログラムの作成方法

この章では、プログラムの作成方法やフローチャート(流れ図)について説明するとともに、第4章のアセンブラを使って簡単な問題を扱ったマシン語プログラムを作成してみます。



5 マシン語プログラムの作成方法

プログラムの作成方法

プログラムの作成にはいろいろな方法があります。そして、面倒な手順や理論があります。初心者の方にこのような方法を説明してもかえってわからなくなってしまうと思いますので、最低こうして作成したほうがよいという方法を説明します。プログラムの作成方法(設計方法)に関する本は多数発刊されていますので、興味のある方はそちらもごらんください。

それでは、プログラムを作成するときの手順を次に示します。

- ①問題をよく理解する。
- ②大まかな処理を決める。
- ③細かい処理を決める。
- ④コーディングする。
- ⑤コーディングした内容を打ち込み、ソース・プログラムをつくる。
- ⑥アセンブルする。
- ⑦マシン語をデバッグする。
- ⑧完成。

この手順だと、マイコンに向かってする作業は⑤からになります。①～④は机上の作業で、マイコンに向かうことはあまりありません。初心者の人はどうしてもマイコンに触れたい心が先に立ち、①～④の作業が雑になったり、ひどい場合はこれらの作業抜きでいきなりマイコンに向かってしまいがちです。これでは、作成時間がかかる割には完成度の低いプログラムがで

きてしまいます。①～④の作業をある程度しっかりすることにより、作成時間も短くなりますし、完成度も高くなります。そして、①～④の作業の間にフローチャートなどが書類として残るので、後になってそのプログラムを修正・改造するときに大変役立ちます。

それでは、この手順を細かくみていきましょう。

①問題をよく理解する(問題解析とプログラムの仕様の決定)

ここでは、何をどうしたいのかをはっきりさせます。たとえば、操作手順や画面の構成、入出力データのチェック方法など細かいことを決めてしまうのです。そして、決めた内容は必ず紙に書いておきます。しかし、ある程度プログラムの概要がわからないと決まらない内容は、②にまわします。

②大まかな処理を決める(プログラムの概要設計)

ここでは①で決めた内容をプログラムにしたとき、大まかにどのような処理をすればよいのかについて考えます。そして考えた処理手順を図式化したフローチャート(Flowchart: ⁽¹⁰⁵⁾流れ図)というものをつくります。この段階では、1つの処理内容を表わすのにたとえば「キーボードからデータX,Y,Zを入力する」とか「A B Cを計算する」などと大まかな表現で書きます。これらはいずれもマシン語に

⑩特に大まかな処理を表わすフローチャートのことをゼネラル・フローチャート(General Flowchart)という。詳しくは5章2(116ページ)参照。

⑪引数ともいう。サブルーチンと呼ぶときに、ある値を取引

する媒介となるレジスタやメモリのこと。たとえば1文字入力ルーチンと呼ぶとき、アキュムレータAに入力された文字コードが入って返ってくるものだとなれば、Aレジスタがパラメータである。

して数10～数100バイトになる内容を表わしています。また、フローチャートで示される各処理について、フローチャートとは別に説明文を書いておくのもよいでしょう。そして、共通に使えるデータ領域やサブルーチンについても考えておきましょう。

③細かい処理を決める（プログラムの詳細設計）

ここでは②で作成したゼネラル・フローチャートから、今度はこれを細かく表わしたディテール・フローチャート（Detail Flowchart）というものを作成します。このフローチャートでは、1つの処理内容を表わすのに、たとえば「HL←HL+32」とか「B←100」のようにマシン語にして1～数バイトになる内容を書きます。また、この段階でサブルーチン名とその処理内容、パラメータやデータ⁽¹⁰⁶⁾を記憶する領域の大きさや形式、ラベル名なども決めます。

④コーディング

この作業は、ディテール・フローチャートなどの内容を元にしてZ80Aのニモニックやアセンブラの擬似命令でプログラムを書く作業です。

⑤コーディングした内容を打ち込む

ここではコーディングされたプログラムをキーボードから打ち込み、ソース・プログラムをつくります。打ち込み終わったソース・プログラムは必ずセーブしておきます。

⑥アセンブル

ソース・プログラムからマシン語プログラムをつくる作業です。アセンブル中にエラーが出るようなら、その行を修正し再度アセンブルを行ないます。

⑦マシン語プログラムのデバッグ

アセンブルによってつくられたマシン語プログラムが正しく動作するかチェックする作業です。サブルーチンなどからチェックしていき最後にプログラム全体をチェックしていった方がよいでしょう。

もしデバッグで大きなミスが発見されたら、前に戻ってフローチャートなどを修正します（最悪の場合①まで戻る）。また、プログラム作成中処理内容の変更などを行なった場合も、前に戻ってフローチャートなどを修正します。

⑧完成

プログラムが完成したら、これまで作成した書類とプログラム・リストを整理・保管しておきましょう。





フローチャート(流れ図)

フローチャートは、コンピュータが処理する手順を図式化したものです。アルゴリズム (Algorithm) ⁽¹⁰⁷⁾ を図で表わしたものがフローチャートであるともいえます。

フローチャートで使う流れ図記号 (Flowchart

Symbol) は JIS (日本工業規格) により30種類ほどのものが決められています (付録1)。

しかし、これらの記号を全部使うわけではありません。初めのうちは、次のものだけで十分でしょう。

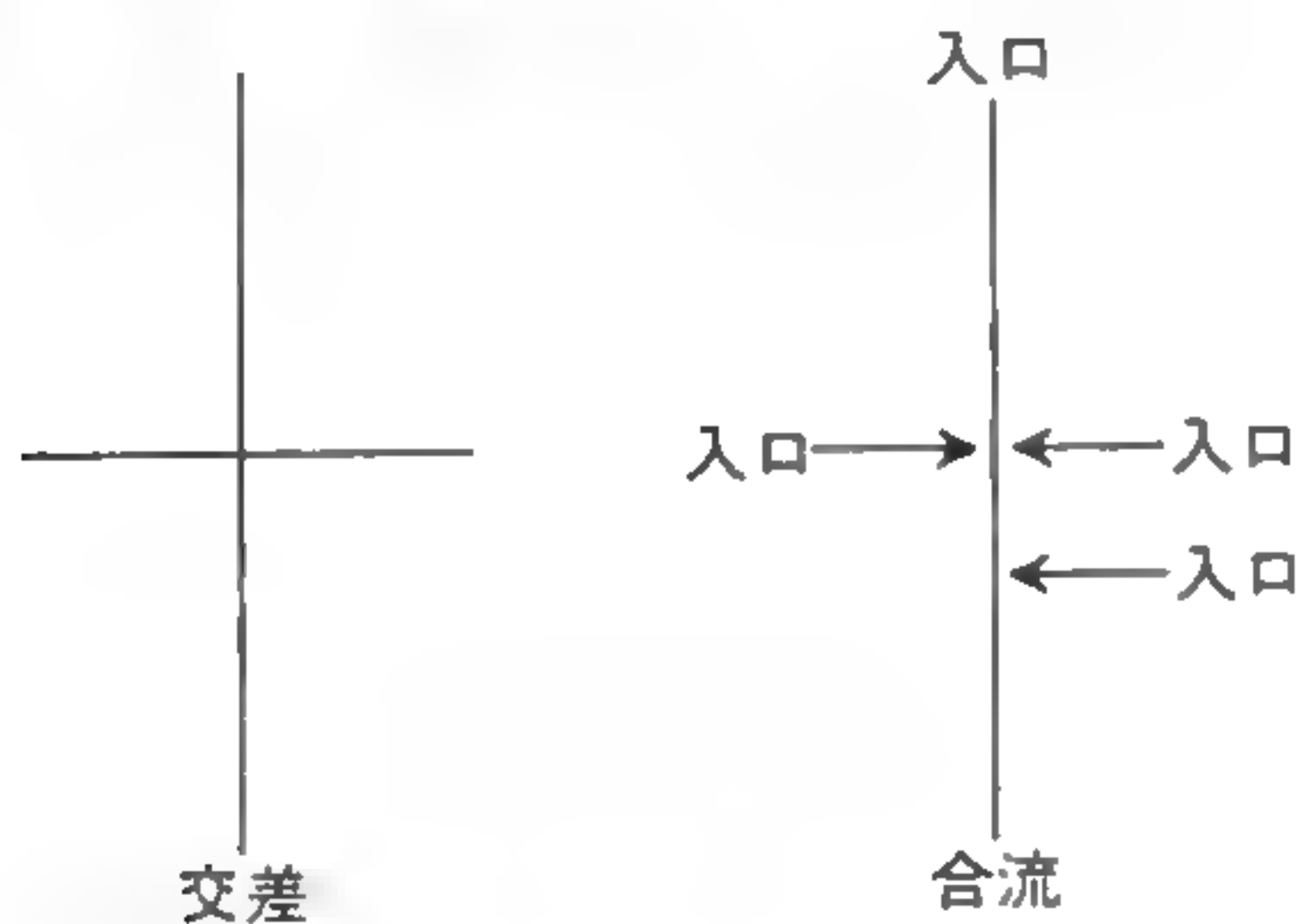
記号(シンボル)	意 味	記号(シンボル)	意 味
	端子 (terminal, interrupt) 開始, 終了, 停止, 中断などを表わします。		のページからの入口を表わします。 この記号は JIS には規定されていませんが、一般によく使われています。
	処理 (process) 種々の処理機能を表わします。		定義済み処理 (predefined process) サブルーチン・コールを表わします。
	判断 (decision) どの経路をとるかの判断を表わします。		入出力 (input/output) 入力または出力を表わします。
	流れ線 (flow line) プログラムの流れを表わします。流れの方向が上から下へ, または左から右への場合は記号として(—)を使ってもかまいません。		書類 (document) プリンタへの印字を表わします。
	結合子 (connector) ほかの場所への出口, またはほかの場所からの入口を表わします。		表示 (display) ディスプレイへの表示を表わします。
	ページ結合子 (offpage connector) ほかのページへの出口, またはほか		手操作入力 (manual input) キーボード等からの入力を表わします。
			注釈 (comment, annotation) 詳しい説明や注意を書くときに使います。

⑩ある問題を解決するための特別な方法・手順のこと。コンピュータでプログラムをつくるということは、対象の問題を解決する手順についてすべての可能性を考え、アルゴリズムを立てることであるといえる。

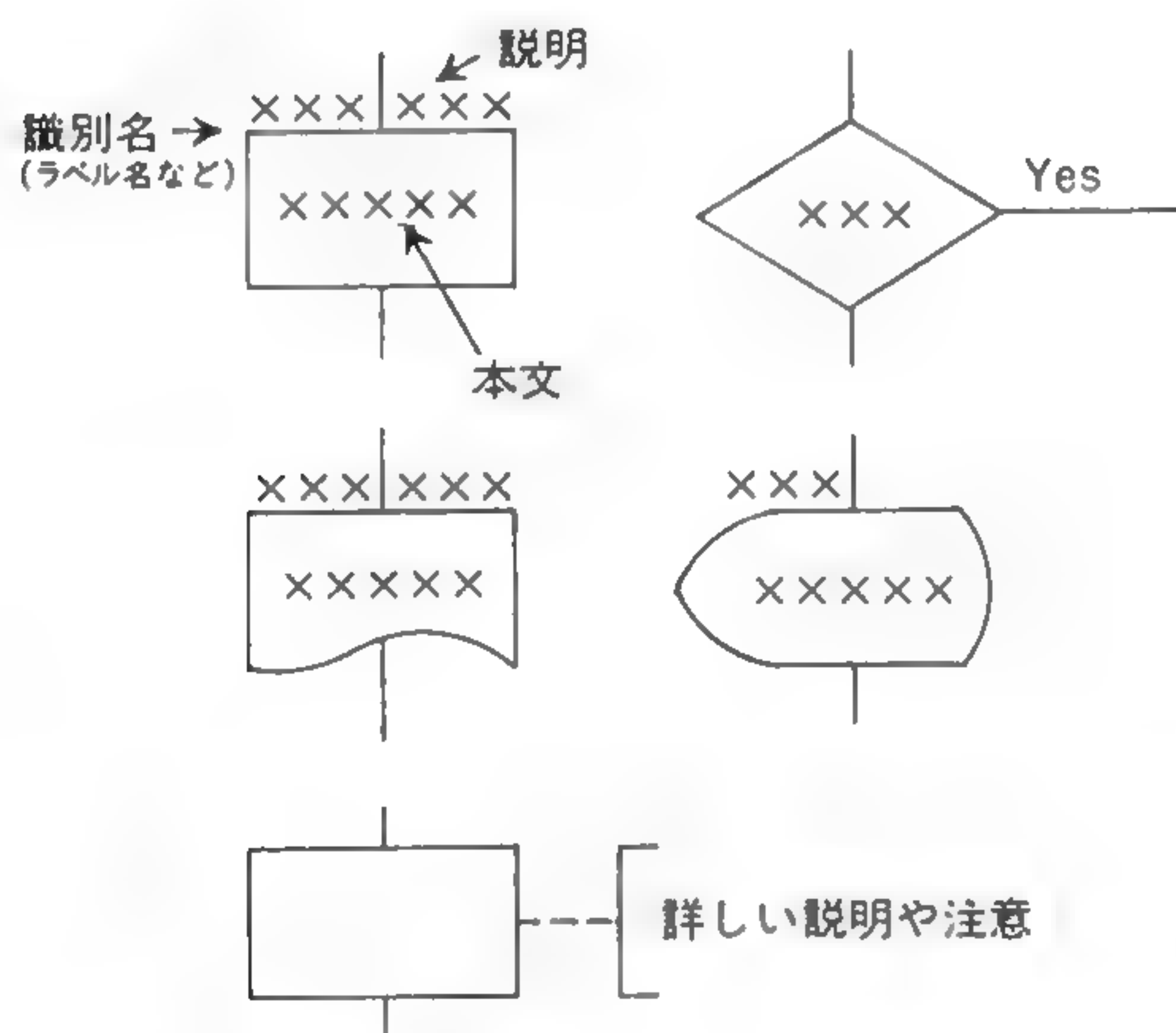
⑪ ROM 内にあって、MSX-BASIC にも利用されている入出力ルーチンのこと。MSX ではほとんど公開されているので、ユーザーが自由に使うことができる。平たくいえば BASIC インタープリタ内の基本サブルーチンである。

フローチャートを書くときの注意として、つぎのようなことがあります。

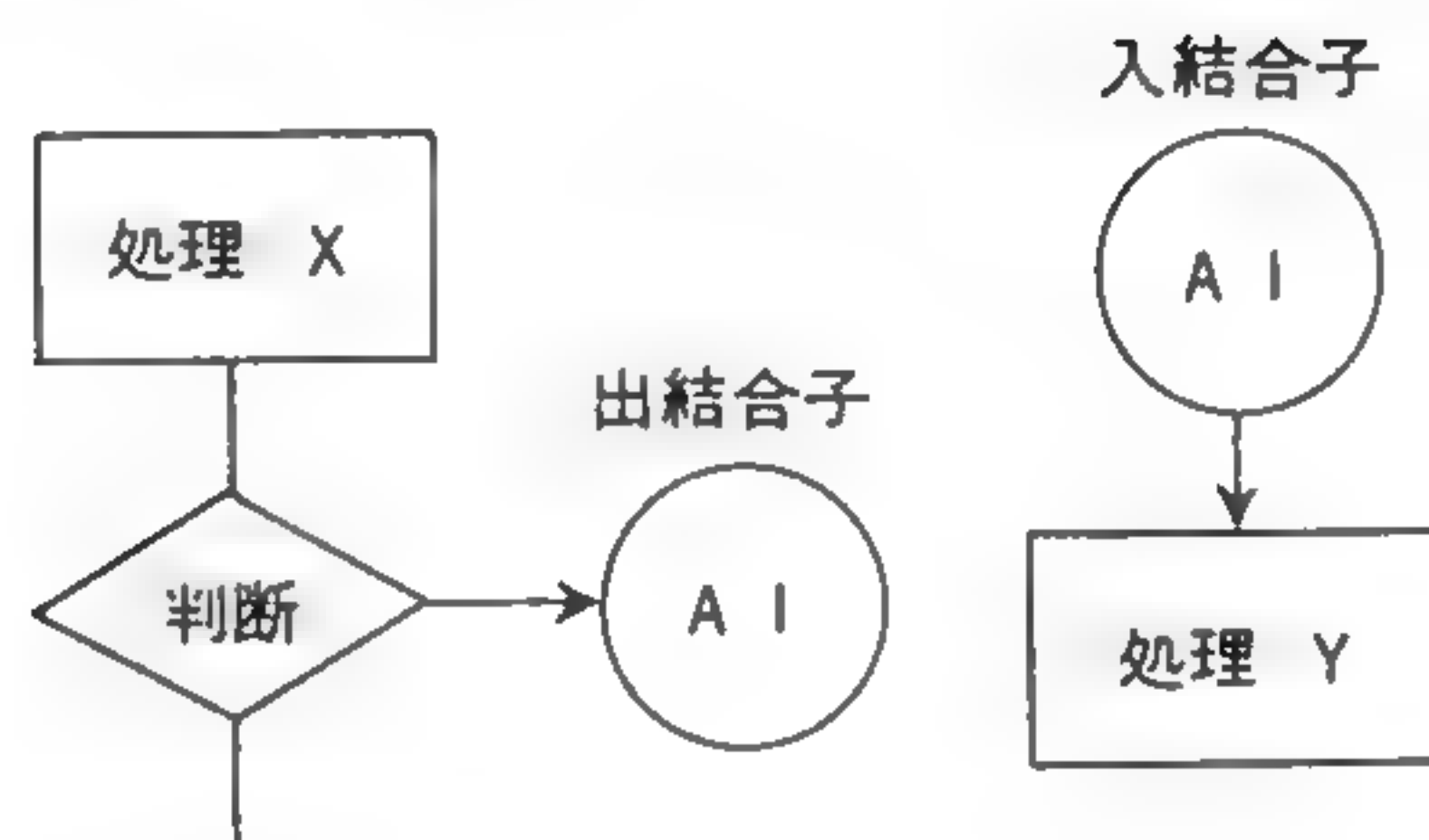
①流れ線の交差と合流の書きかた



②識別名と説明の書きかた

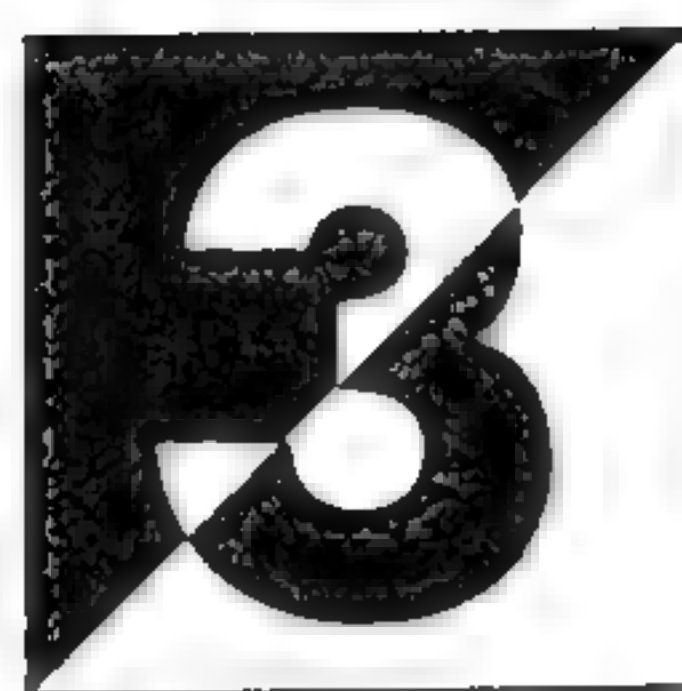
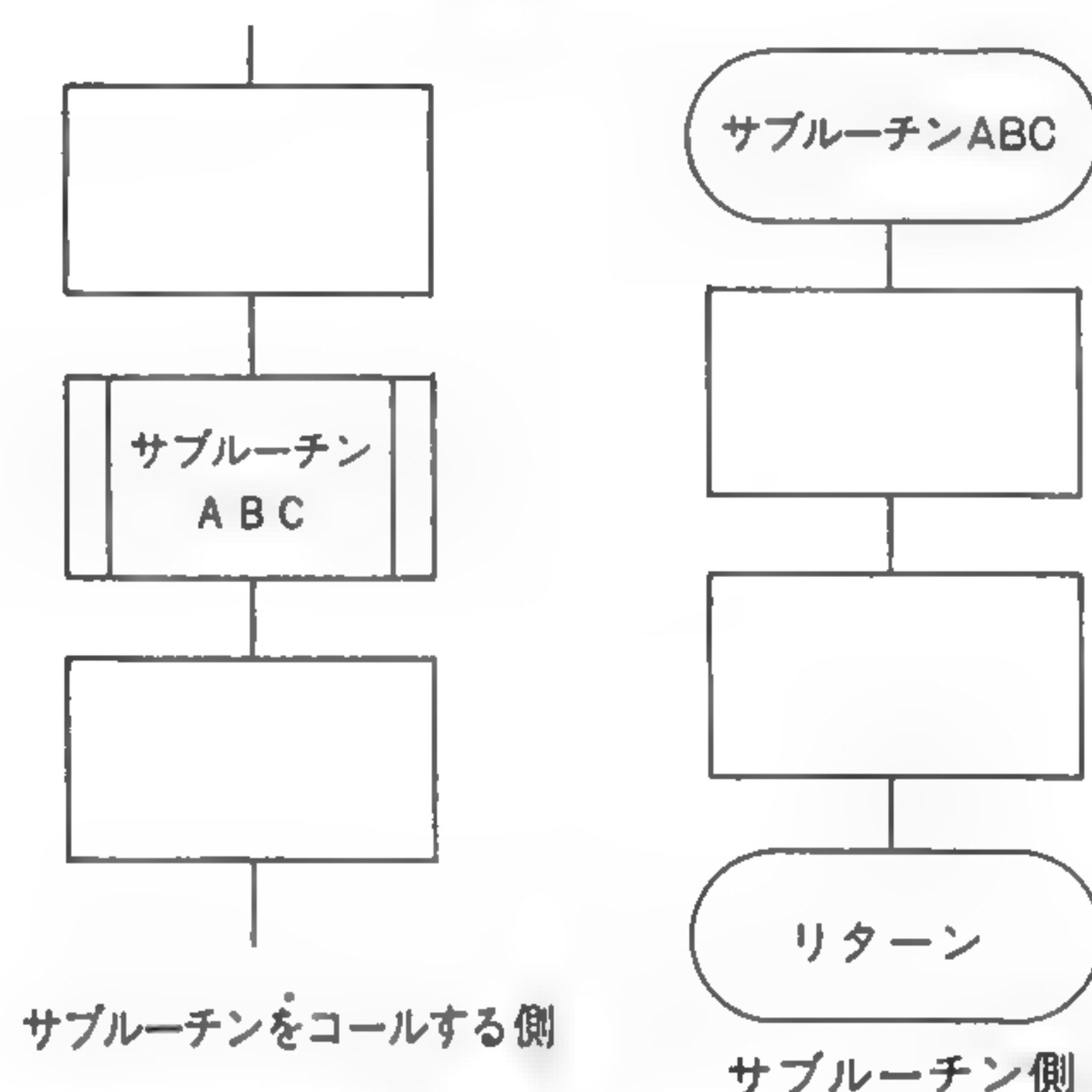


③結合子の使いかた



対応する出結合子と入結合子には同じ文字や数字、名前などを記入します。

④サブルーチンの表わしかた



5 マシン語プログラムの作成方法

マシン語プログラムの作成例

ここでは5章1で説明した作成手順にしたがって、次のようなプログラムを作成し、実際に実行してみます。

〔問題〕

- ・プログラム名は「サンプル・プログラムNo2」。
- ・キーボードから20文字の数字(0～9)を入力し、どの数字が何文字入力されたか表示する。数字以外の文字は無視すること。

①初めに、問題の内容からどのようなプログ

ラムにするか仕様を決めます。

プログラムの仕様

1. キーボードからの入力と画面への表示はROMOS⁽¹⁰⁸⁾を使う。
2. 入力できる文字は0～9の10種類。キーボードから入力しただけでは画面に表示されないの、入力された文字は画面にも表示する。
3. 入力文字数を記憶する領域の大きさと形式を決める。

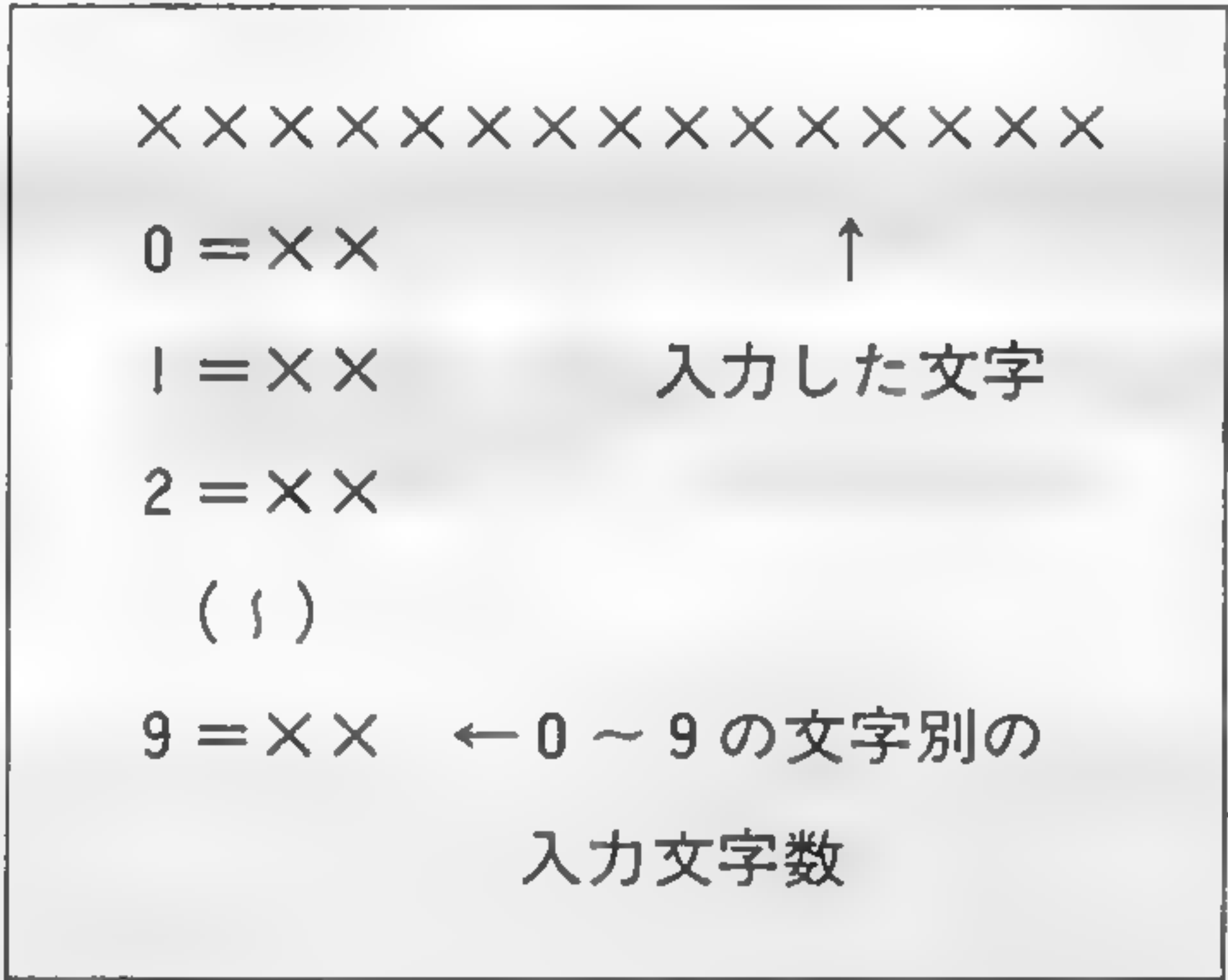
・入力 は 20 文字なので、1 種類の文字は最大 20 回入力される。

・入力文字数 (0 ~ 20) を表現する方法を考える。BCD なら 1 バイトで 2 桁 (00 ~ 99) ，バイナリ (2 進数) なら 1 バイトで 0 ~ 255 まで表わせるので、バイナリ / BCD どちらでもよい。ただ表示するとき 10 進数に変換することを考えて、変換しやすい BCD を使うことに決める。

・入力文字数を記憶する領域は、文字が 10 種類なので 10 種類 × 1 バイト = 10 バイト必要だと考えられる。

4. このプログラムは BASIC から ⁽¹⁰⁹⁾USR 関数で実行する。プログラムの実行アドレスは D300 番地とする。

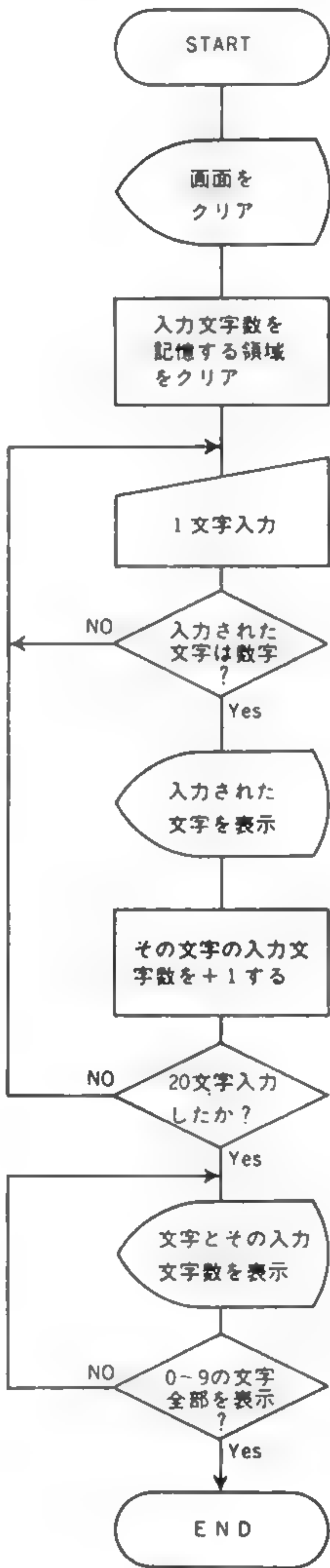
5. プログラムを実行したとき、初めに画面をクリアし、次のように入出力することとする。



②プログラムの仕様が決まったら、次にゼネラル・フローチャートをつくります (図 5-1) 。

③ゼネラル・フローチャートをつくり終えたら、次はプログラムの詳細な設計をします。ここでは、図 5-2、図 5-5 のようなディテール・フローチャート、図 5-3 のようなデータ領域の内容、図 5-4 のようなサブルーチンの一覧表などもつくります。

図 5-1 サンプル・プログラム No.2 のゼネラル・フローチャート



⑨ BASIC からマシン語サブルーチンを呼ぶための BASIC 命令。関数の形式でマシン語ルーチンを呼ぶ。詳しくは MSX-BASIC のマニュアルを参照。

図5-2 メイン・ルーチンのディテール・フローチャート

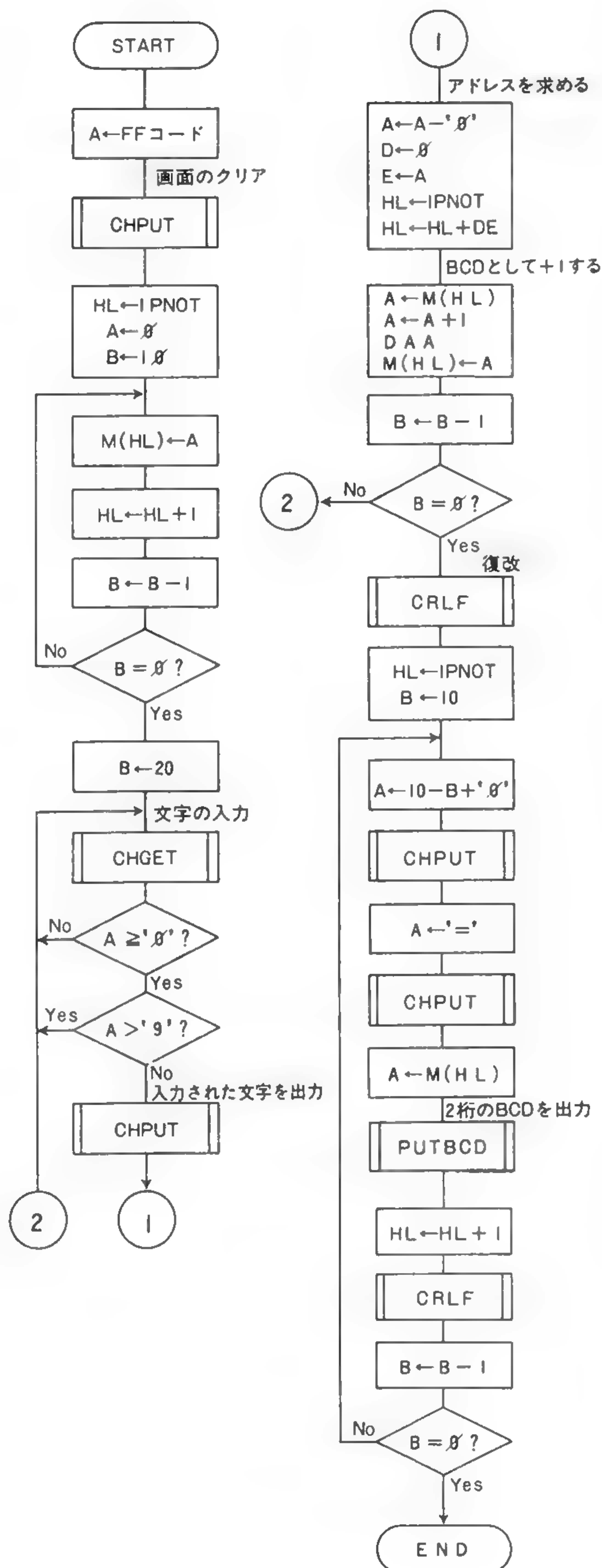


図5-3 データ領域の内容

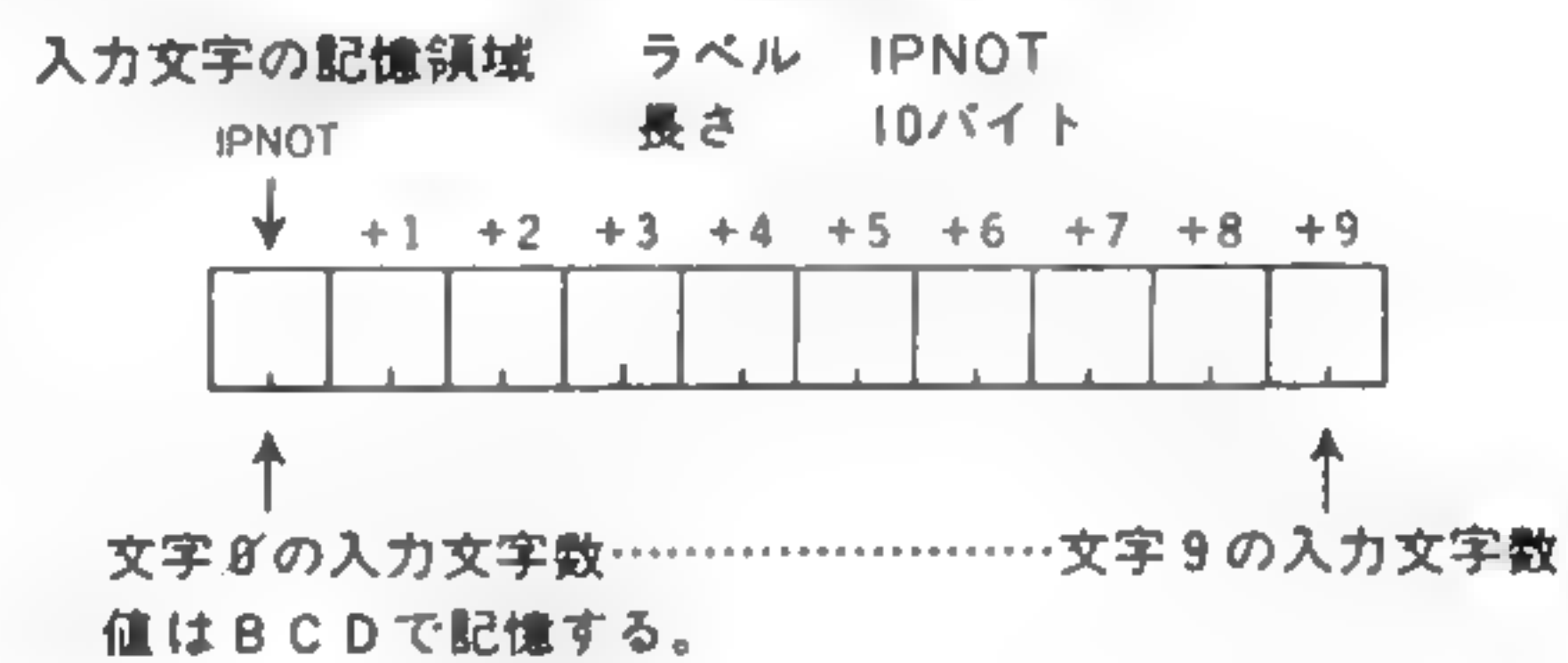
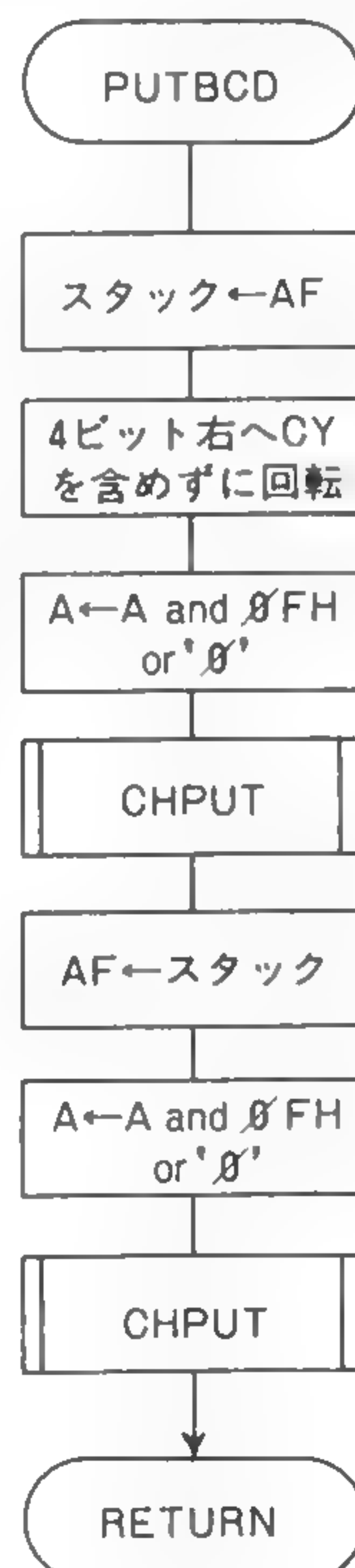


図5-4 サブルーチン一覧表

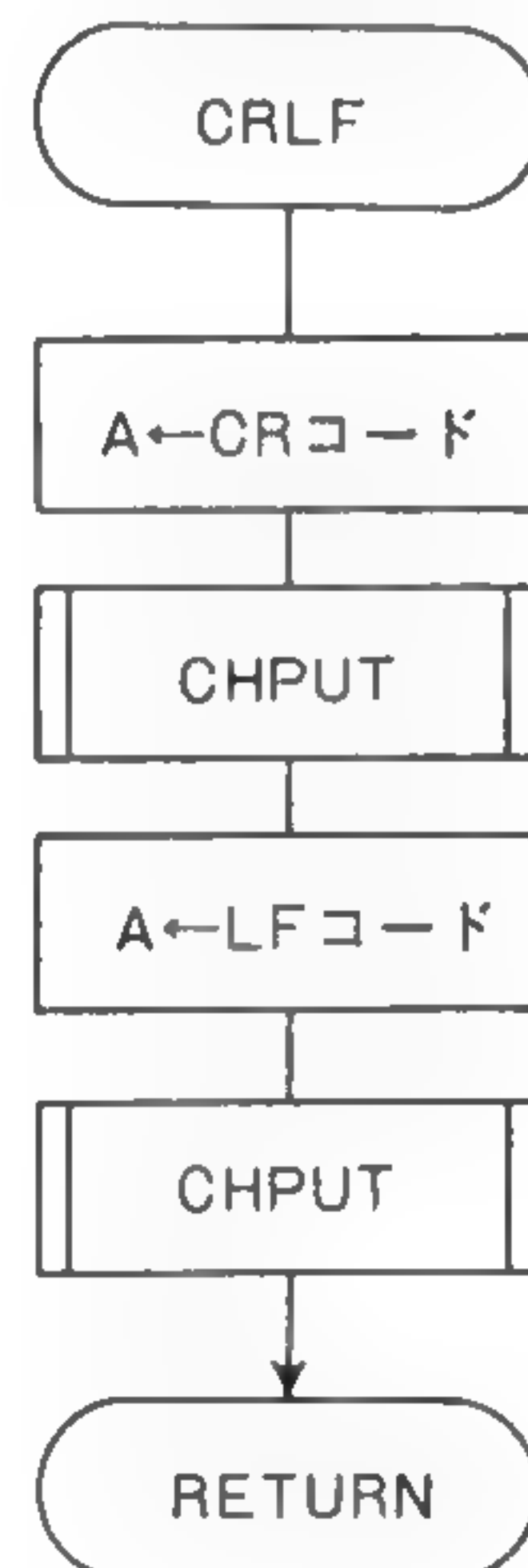
ラベル名	パラメータ	結果	変化するレジスタ	内 容
CHGET	なし	A=文字コード	A F	キーボードから1文字入力する。(ROMOS, コールアドレス=9FH)
CHPUT	A = 文字コード	なし	なし	画面に1文字出力する。(ROMOS, コールアドレス=0A2H)
PUTBCD	A = BCD 2桁	なし	A F	画面にBCD 2桁を10進数に変換して出力する。
CRLF	なし	なし	A F	画面にCR, LFコードを出力する(復改する)。

図5-5 サブルーチンのディテール・フローチャート

・サブルーチンPUTBCD



・サブルーチンCRLF



④及び⑤コーディングし、ソース・プログラムをつくれます。ソース・プログラムはリスト5-1のようになりました。これは、ディテール・フロ

ーチャートの内容と一致します。

⑥アセンブルしてできたのがリスト5-2のプログラム・リストとラベル・リストです。

リスト 5-1 ソース・プログラム

1000 ;	1480 ;
1010 ; ** Sample Program No.2 **	1490 ; call crlf
1020 ;	1500 ; ld hl,ipnot
1030 ; org 0d400h	1510 ; ld b,10
1040 ;	1520 ; loop2:
1050 ; ** MSX ROMOS / エントリー・アドレス **	1530 ; ld a,10
1060 ;	1540 ; sub b
1070 ; chget equ 009fh	1550 ; add a,'0'
1080 ; chput equ 00a2h	1560 ; call chput
1090 ;	1570 ; ld a,'='
1100 ; ** CTRL コード **	1580 ; call chput
1110 ;	1590 ; ld a,(hl)
1120 ; ff equ 0ch	1600 ; call putbcd
1130 ; cr equ 0dh	1610 ; inc hl
1140 ; lf equ 0ah	1620 ; call crlf
1150 ;	1630 ; djnz loop2
1160 ; ** メイン ルーチン **	1640 ; ret
1170 ;	1650 ;
1180 ; start:	1660 ; ** サブ ルーチン **
1190 ; ld a,ff	1670 ;
1200 ; call chput	1680 ; putbcd:
1210 ;	1690 ; push af
1220 ; ld hl,ipnot	1700 ; rrca
1230 ; xor a	1710 ; rrca
1240 ; ld b,10	1720 ; rrca
1250 ; mclr:	1730 ; rrca
1260 ; ld (hl),a	1740 ; and 0fh
1270 ; inc hl	1750 ; or '0'
1280 ; djnz mclr	1760 ; call chput
1290 ;	1770 ; pop af
1300 ; ld b,20	1780 ; and 0fh
1310 ; loop1:	1790 ; or '0'
1320 ; call chget	1800 ; call chput
1330 ; cp '0'	1810 ; ret
1340 ; jr c,loop1	1820 ;
1350 ; cp '9'+1	1830 ; crlf:
1360 ; jr nc,loop1	1840 ; ld a,cr
1370 ; call chput	1850 ; call chput
1380 ; sub '0'	1860 ; ld a,lf
1390 ; ld d,0	1870 ; call chput
1400 ; ld e,a	1880 ; ret
1410 ; ld hl,ipnot	1890 ;
1420 ; add hl,de	1900 ; ** ワーク エリア **
1430 ; ld a,(hl)	1910 ;
1440 ; add a,1	1920 ; ipnot: defs 10
1450 ; daa	1930 ;
1460 ; ld (hl),a	1940 ; end
1470 ; djnz loop1	

リスト 5-2 プログラム・リスト

1000:		;	
1010:		;	** Sample Program No.2 **
1020:		;	
1030:	D400	org	0d400h
1040:		;	
1050:		;	** MSX ROMOS / エントリーアドレス **
1060:		;	
1070:	009F =	chget	equ 009fh
1080:	00A2 =	chput	equ 00a2h
1090:		;	
1100:		;	** CTRL コード **
1110:		;	
1120:	000C =	ff	equ 0ch
1130:	000D =	cr	equ 0dh
1140:	000A =	lf	equ 0ah
1150:		;	
1160:		;	** メイン ルーチン **
1170:		;	
1180:	D400	start:	
1190:	D400 3E0C	ld a,ff	} 画面をクリアする。
1200:	D402 CDA200	call chput	
1210:		;	
1220:	D405 216FD4	ld hl,ipnot	} ipnotから10バイト、ゼロにする。
1230:	D408 AF	xor a	
1240:	D409 060A	ld b,10	
1250:	D40B	→mclr:	
1260:	D40B 77	ld (hl),a	
1270:	D40C 23	inc hl	} 20文字入力する まで繰り返す。
1280:	D40D 10FC	djnz mclr	
1290:		;	
1300:	D40F 0614	ld b,20	
1310:	D411	→loop1:	
1320:	D411 CD9F00	call chget	……キーボードから文字入力する。
1330:	D414 FE30	cp '0'	} 入力された文字が "0"~"9" 以外なら loop 1にジャンプする。
1340:	D416 38F9	←jr c,loop1	
1350:	D418 FE3A	cp '9'+1	
1360:	D41A 30F5	←jr nc,loop1	
1370:	D41C CDA200	call chput	……入力された文字を画面に出力する。
1380:	D41F D630	sub '0'	} 入力された文字に対応 する入力文字数を記憶 しているアドレスを HLに求める。
1390:	D421 1600	ld d,0	
1400:	D423 5F	ld e,a	
1410:	D424 216FD4	ld hl,ipnot	
1420:	D427 19	add hl,de	
1430:	D428 7E	ld a,(hl)	
1440:	D429 C601	add a,1	} HLが示す内容をBCD で+1する。
1450:	D42B 27	daa	
1460:	D42C 77	ld (hl),a	
1470:	D42D 10E2	djnz loop1	


```

1480:                                     ;
1490: D42F CD64D4      call crlf-----画面に CR, LF コードを出力する。
1500: D432 216FD4      ld hl,ipnot
1510: D435 060A        ld b,10
1520: D437
1530: D437 3E0A        loop2:
1540: D439 90          ld a,10
1550: D43A C630        sub b
1560: D43C CDA200      add a,'0'
1570: D43F 3E3D        call chput
1580: D441 CDA200      ld a,'='
1590: D444 7E          call chput
1600: D445 CD4FD4      ld a,(hl)
1610: D448 23          call putbcd
1620: D449 CD64D4      inc hl
1630: D44C 10E9        call crlf
1640: D44E C9          djnz loop2
1650:                                     ;
1660:                                     ; ** サブ ルーチン **
1670:                                     ;
1680: D44F      putbcd:
1690: D44F F5          push af
1700: D450 0F          rrca
1710: D451 0F          rrca
1720: D452 0F          rrca
1730: D453 0F          rrca
1740: D454 E60F      and 0fh
1750: D456 F630      or '0'
1760: D458 CDA200      call chput
1770: D45B F1          pop af
1780: D45C E60F      and 0fh
1790: D45E F630      or '0'
1800: D460 CDA200      call chput
1810: D463 C9          ret
1820:                                     ;
1830: D464      crlf:
1840: D464 3E0D      ld a,cr
1850: D466 CDA200      call chput
1860: D469 3E0A      ld a,lf
1870: D46B CDA200      call chput
1880: D46E C9          ret
1890:                                     ;
1900:                                     ; ** ワーク エリア **
1910:                                     ;
1920: D46F      ipnot: defs 10 -----文字 "0"~"9" に対応する入力
1930:                                     ;                               文字数を記憶する領域。
1940: D479      end

```

画面に文字を出力する。
1回目のループで文字"0",
10回目のループで文字"9"
が出力される。

画面に "=" を出力する。

メモリに記憶されている入力
文字数を画面に出力する。

このループを
10回繰り返す。

レジスタ A にセットされてきた B C D 2 桁を
2 桁の 10 進数として画面に出力するサブルーチン

画面に CR, LF コードを出力するサブルーチン

リスト 5-2 ラベル・リスト

009F	CHGET	00A2	CHPUT	000D	CR	D464	CRLF
000C	FF	D46F	IFNOT	000A	LF	D411	LOOP1
D437	LOOP2	D40B	MCLR	D44F	PUTBCD	D400	START

⑦モニタでオブジェクトをロードします。オブジェクトはメモリのD400番地からD478番地にロードされます。初めにモニタでサブルーチンPUTBCDとCRLFをチェックし、

次にメイン・ルーチンをチェックします。
⑧操作例5-1は完成したプログラムの実行例です。

操作例 5-1 リスト5-2のプログラムの実行例

```
DEF USR=&HD400
A=USR(0)
```

```
13679654324509845835...キーボードより入力した文字
0=01
1=01
2=01
3=03
4=03
5=04
6=02
7=01
8=02
9=02
Ok
```

文字別の入力された文字数。
例えば文字0は1回、文字5は4回入力されたことを示す。

最 後 に

以上、MSXによるマシン語プログラムの作成に最低必要と思われることについて述べてきましたが、本格的に勉強なさる方は、下に記載した参考文献もあわせてごらんください。特に、マイコン以外のコンピューター一般について勉強したい方は、オーム社より発行されている「ソフトウェアの知識」「ハードウェアの知識」をおすすめします。

また、Z80A CPUについて詳しくお知りになりたい方は、NECから発行されている「μCOM-82 ユーザーズ・マニュアル」やシャープ、サイログなど各社から発行されているマニュアルをごらんください。そして、とりあえず必要ないかもしれませんが、JISの規格表や用語辞典なども持っているとはプログラム作成のときに役立ちます。

それでは、皆さんの御健闘を祈り本書を終わらせていただきます。

〈参考文献〉

- (1) JIS ハンドブック 情報処理 1981 日本規格協会
- (2) μCOM-82 ユーザーズ・マニュアル NEC IEM-616 AUG-18-78
- (3) 矢田光治著 ソフトウェアの知識 (第2版) オーム社 1975
- (4) マイクロプロセッサ教育研究会編 マイコン用語辞典 電波新聞社 1978
- (5) 渡辺一郎, 平原英夫共著 コンピュータ用語辞典 富士書房 1980
- (6) 桜田幸嗣, 田村明史著 PC-6001 マシン語入門 アスキー 1983
- (7) 大須賀節雄, 近谷英昭共著 ハードウェアの知識 オーム社 1970
- (8) MSX システムの全貌 月刊アスキー 1983年 8月号
- (9) MSX 最新情報ファイル 月刊アスキー 1983年 10月号~11月号

Appendix

Appendix 1 ————— 付- 1

JIS C6270-1975「情報処理用流れ図記号」

Appendix 2 ————— 付- 6

MSXマシンのキャラクタ・コード表

Appendix 3 ————— 付- 7

Z80Aインストラクション一覧表

Appendix 4 ————— 付-18

マシン語・モニタ対応表

情報処理用流れ図記号 JIS C6270-1975

1. 適用範囲

この規格は、情報処理用流れ図記号 (flowchart symbol) とその用法について規定する。

＜備考＞

流れ図記号は、情報処理系における操作の系列、データの流れ、作業の進行などを表現するためのものである。したがって、絵や図を用いて情報処理系を描写する絵画的なものについては規定しない。

2. 用法の一般事項

①流れ図 (flowchart) における流れ (flow) の方向は、原則として、左から右へ、上から下へとする。流れの方向がこれに合わないときには、流れを示す矢印を用いなければならない。

なお、矢印を用いたほうがわかりやすいときには、これを用いることが望ましい。

②流れ線 (flow line) は互いに交差してもよい。この場合には、これらの間に互いに論理的関係はないものとする。


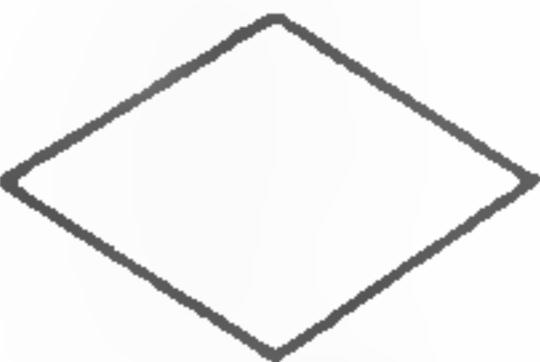



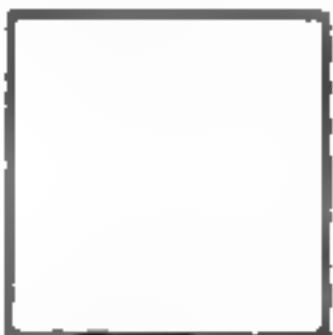


③二つ以上の流れ線を集めて、一つの流れ線にして出してもよい。









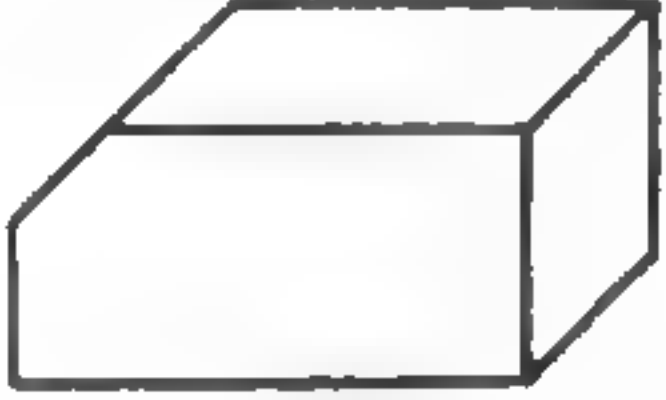




④記号は、直ちに識別できないほどまでに、形を変えたり回転したりしてはならない。

＜備考＞

記号の大きさや縦横の比率については特に規定しない。記号は、実線によることを原則とするが、特に必要な場合には、破線などを使ってもよい。

3. 流れ図記号 流れ図記号は、次のとおりとする。

番 号	記 号	名 称	意 味
1		処 理 (process)	あらゆる種類の処理機能を表す。
2		判 断 (decision)	いくつかの択一的径路のうち、どの径路をとらせるかを決める判断、又はスイッチ式の操作を表す。
3		準 備 (preparation)	スイッチの設定、指標レジスタの変更、ルーチンの初期値の設定など、プログラム自身を変えるための命令又は命令群の修飾を表す。
4		定義済み処理 (predefined process)	サブルーチンなど、別の場所で定義されている命令群、又はいくつかの操作からなる命令された処理過程を表す。
5		手作業 (manual operation)	機械的な手段を用いない手作業によるオフラインの処理過程を表す。
6		補助操作 (auxiliary operation)	中央処理装置で直接に制御されていない装置によって行うオフライン操作を表す。
7		組合せ (merge)	記録の二つ以上の集合を一つの集合に組み合わせることを表す。
8		抽出し (extract)	いくつかの記録からなる一つの集合の中から、特定のいくつかの集合を取り分けることを表す。

番 号	記 号	名 称	意 味
9		照 合 (collate)	組み合わせたり抜き出したりしながら、記録の二つ以上の集合から二つ以上の集合を作り出すことを表す。
10		分 類 (sort)	集合中の記録をある定められた順序に並べかえることを表す。
11		手操作入力 (manual input)	オンラインけん盤、スイッチ、押しボタンなどを手で操作して、処理中に情報を入れる入力機能を表す。
12		入出力 (input/output)	情報を処理可能にする入力機能、又は処理済みの情報を記録する出力機能を表す。
13		オンライン記憶 (online storage)	磁気テープ、磁気ドラム、磁気ディスクなど、あらゆる種類のオンライン記憶を利用する入出力機能を表す。
14		オフライン記憶 (offline storage)	情報を任意のオフラインの記録媒体に記憶する機能を表す。
15		書 類 (document)	書類を媒体とする入出力機能を表す。
16		紙カード (punched card)	紙カードを媒体とする入出力機能を表す。
17		カードの組 (deck of cards)	紙カードの集まりを表す。
18		カードのファイル (file of cards)	紙カード上の関連ある記録の集まりを表す。
19		紙テープ (punched tape)	紙テープを媒体とする入出力機能を表す。
20		磁気テープ (magnetic tape)	磁気テープを媒体とする入出力機能を表す。
21		磁気ドラム (magnetic drum)	磁気ドラムを媒体とする入出力機能を表す。

番 号	記 号	名 称	意 味
22		磁気ディスク (magnetic disk)	磁気ディスクを媒体とする入出力機能を表す。
23		磁 心 (core)	磁心を媒体とする入出力機能を表す。
24		表 示 (display)	処理中に、オンライン表示燈、映像表示装置、操作卓の印字機、作図機などを通じて、情報を人間が利用できるように表示する入出力機能を表す。
25		流れ線 (flow line)	記号を結びつける機能を表す。 用法は2.①による 流れ線の交差は、2.②による。 流れ線の合流は、2.③による。
26		並行処理 (parallel mode)	二つ以上の同時操作の始まり、又は終わりを表す（左図には流れ線は示してない）。
27		通 信 (communication link)	情報を通信線で伝達する機能を表す。
28		結合子 (connector)	流れ図のほかの場所への出口、又はほかの場所からの入口を表す。
29		端 子 (terminal, interrupt)	開始、終了、停止、遅延、中断など、流れ図の端子を表す。
30		注 釈 (comment, annotation)	明りょうにするために、説明又は注意を加える機能を表す。

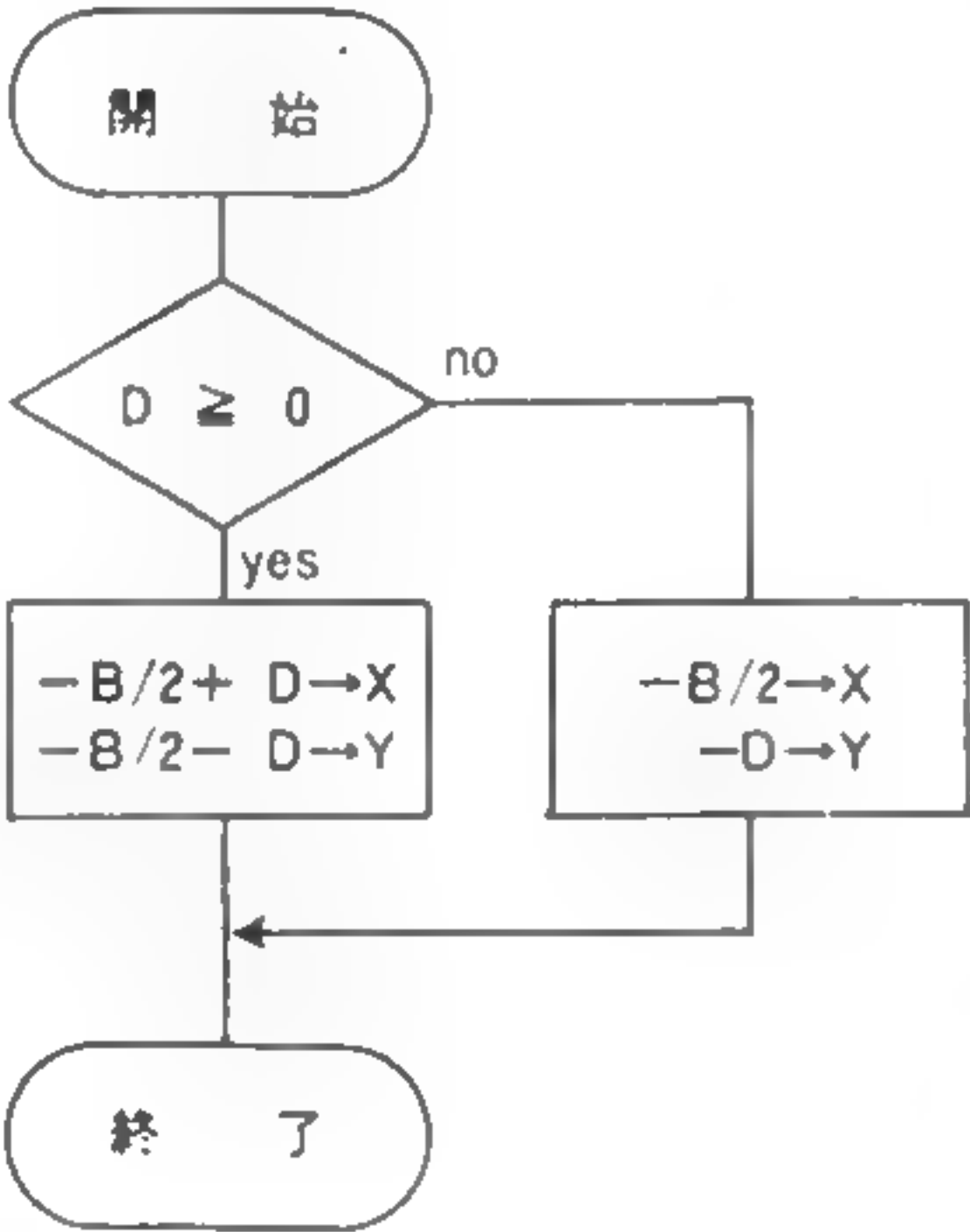
4. 流れ図中の記述

①本 文 流れ図記号に付ける本文(text)は、流れの方向にかかわらず、左から右へ、上から下へ読むように書く。

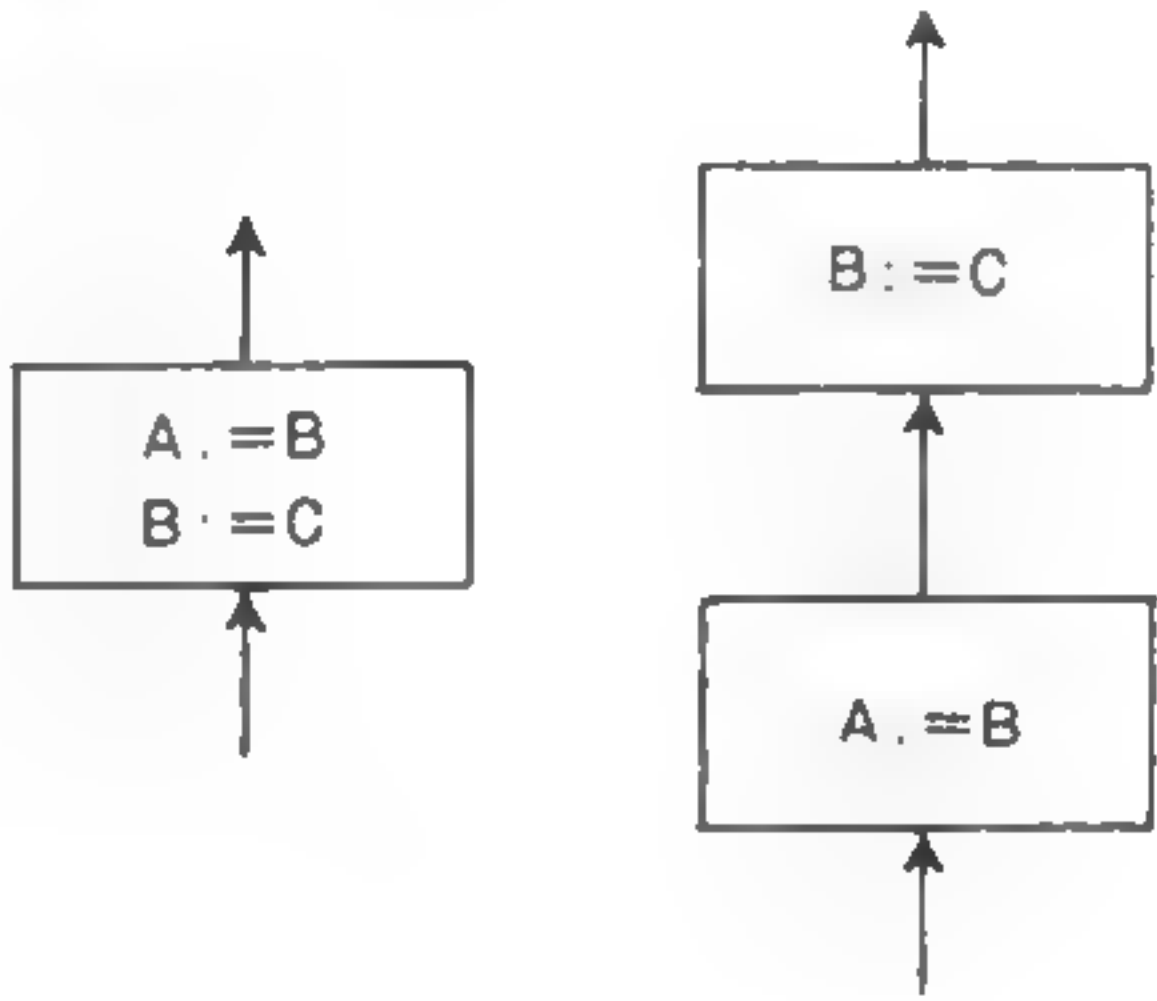
〈備考〉

本文は、可能な限り記号の内部に記入することが望ましい。

例：図- 1



例：図- 2



※左図の処理は右図のように解釈する。

②**識別名** 流れ図記号に付ける識別名 (symbol name) は、プログラムなどを参照できるように記号に名前を付けるもので、記号の左上に書く。

③**説明** 流れ図記号に付ける説明 (symbol description) は、流れ図が表す系の中での、この記号の機能をよりよく理解できるようにするために記載する前後参照など、あらゆる情報であって、記号の右上に書く。

5. 結合子

①**出結合子** 流れ線を中断する点を表わす結合子を、出結合子 (outconnector) と呼ぶ。

②**入結合子** 中断された流れ線を再開する点を表わす結合子を、入結合子 (inconnector) と呼ぶ。

③**結合** 出結合子及び対応する入結合子の中に同じ文字、数字、名前などの識別子 (identifier) を記入し、これらの結合を表すものとする。

6. 横線

①**機能** 流れ図記号の中に書きこまれた横線は、この記号に関する、より詳細な表現が同じ流れ図中のほかの場所に記載してあることを示す。

〈備考〉

定義済み処理では、横線の書きこまれた記号とは異なり、より詳細な表現が同じ流れ図にある必要はない。

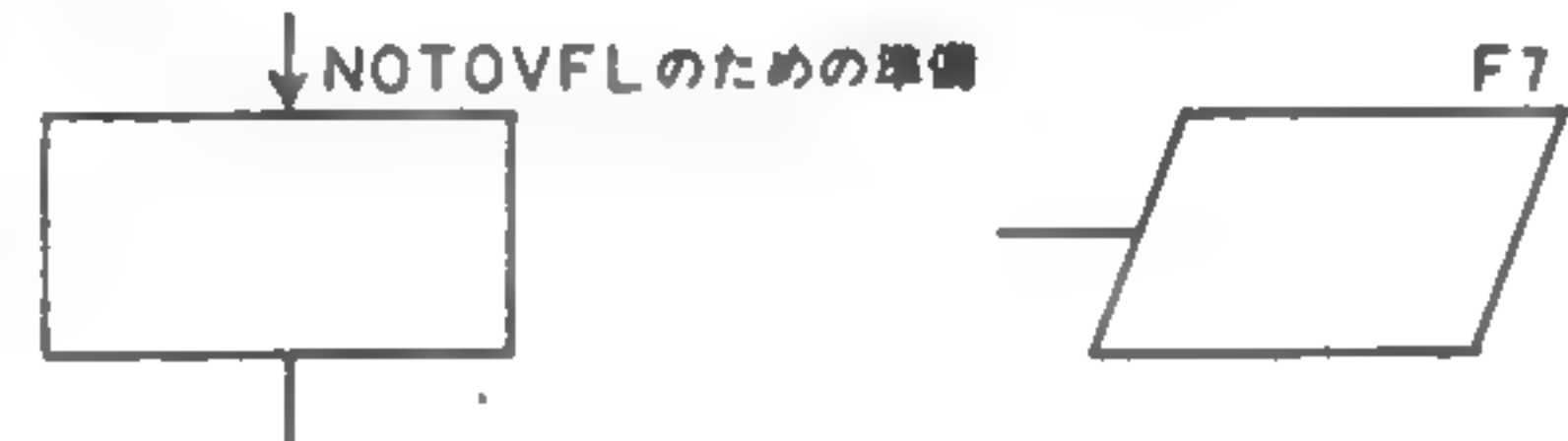
②**書き方** 横線は、流れ図記号の上縁に近い内側に水平に記入する。また、この記号を詳細に表現した場所に対する識別子を、記号の上縁と横線との間に記入する。

③**詳細表現の参照** 横線の書きこまれた流れ図記号 (striped symbol) を詳細に表現した場所では、その始まりと終わりとに端子を用いる。始まりの端子の中には、6.②に従って記入した識別子と同じ識別子を記入する。

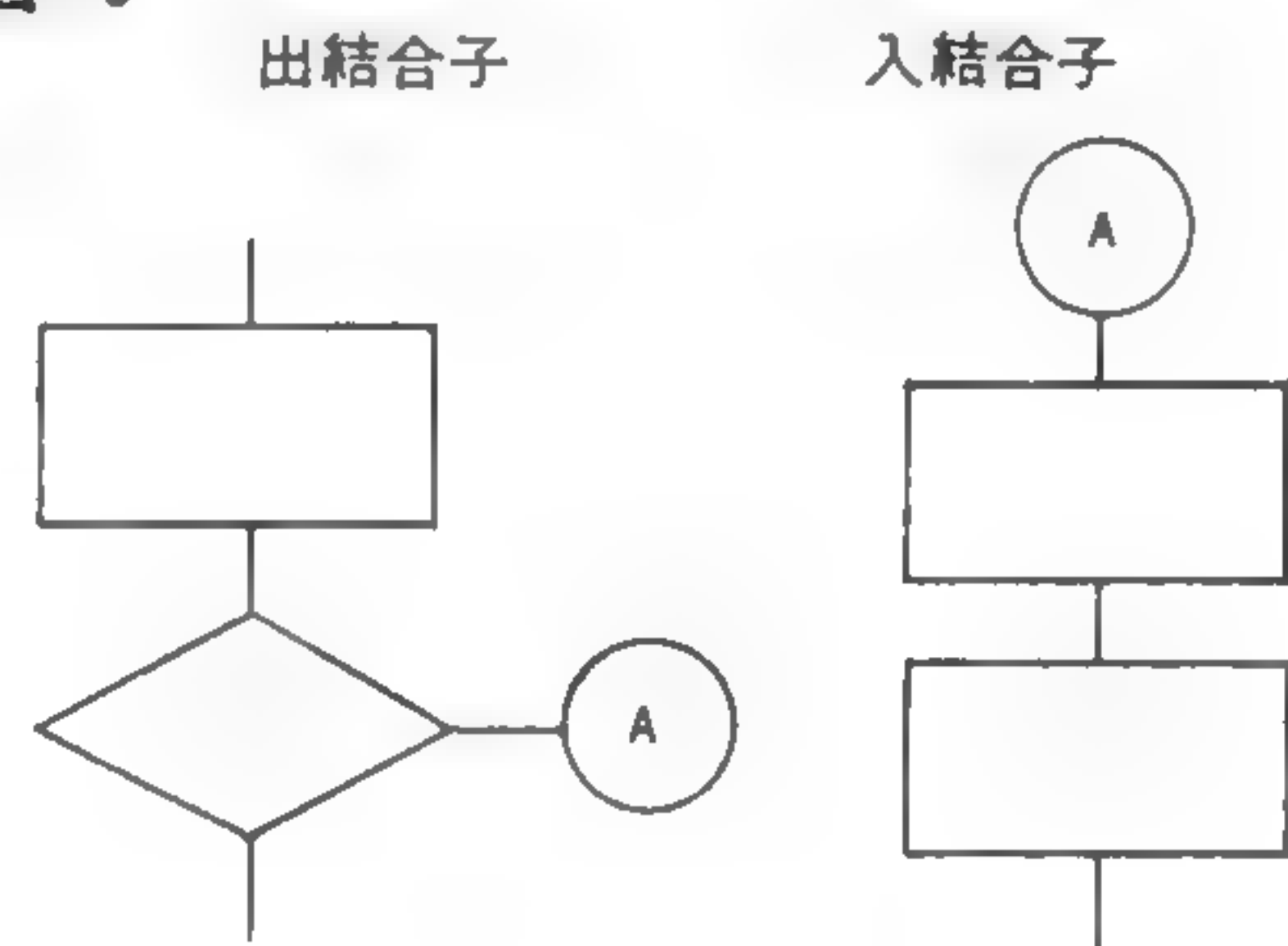
例：図-3



例：図-4

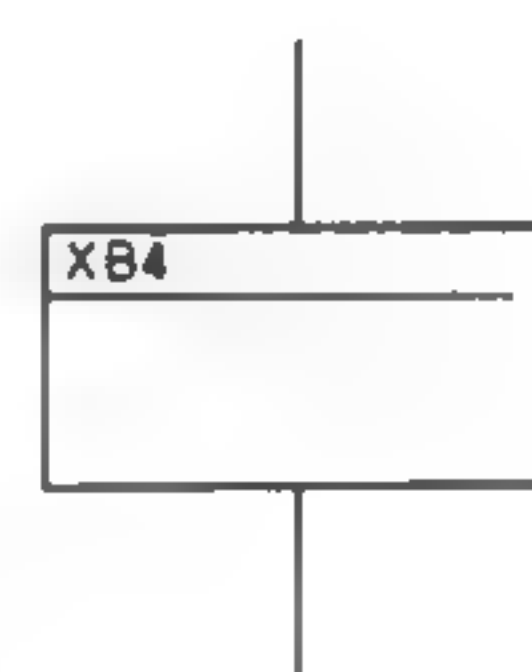


例：図-5

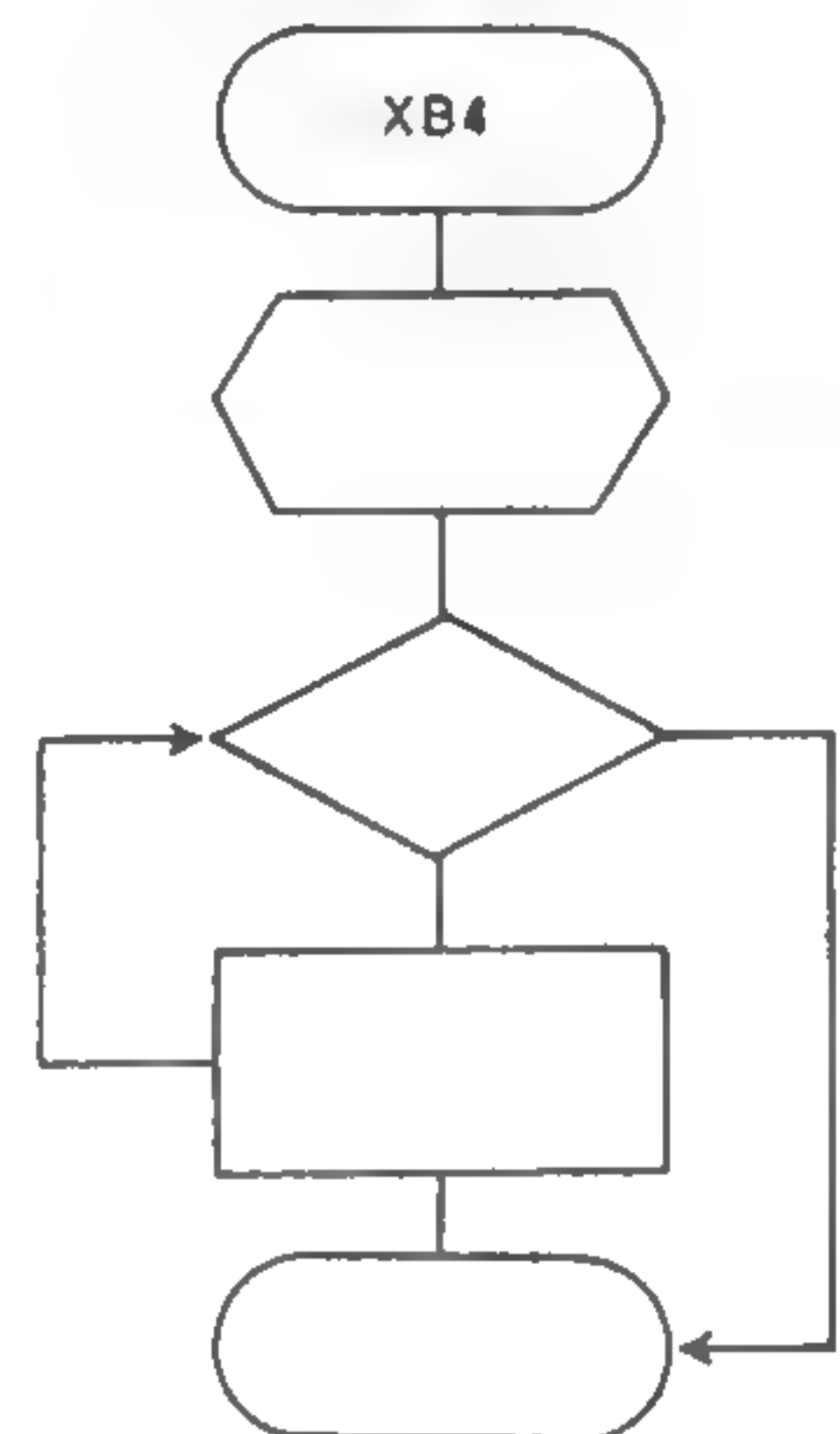


例：図-6

横線の書きこまれた記号



詳細な表現



※横線を記号の左右の縁につくまで十分に延ばすかどうかは、規定しない。

7. 二つ以上の出口

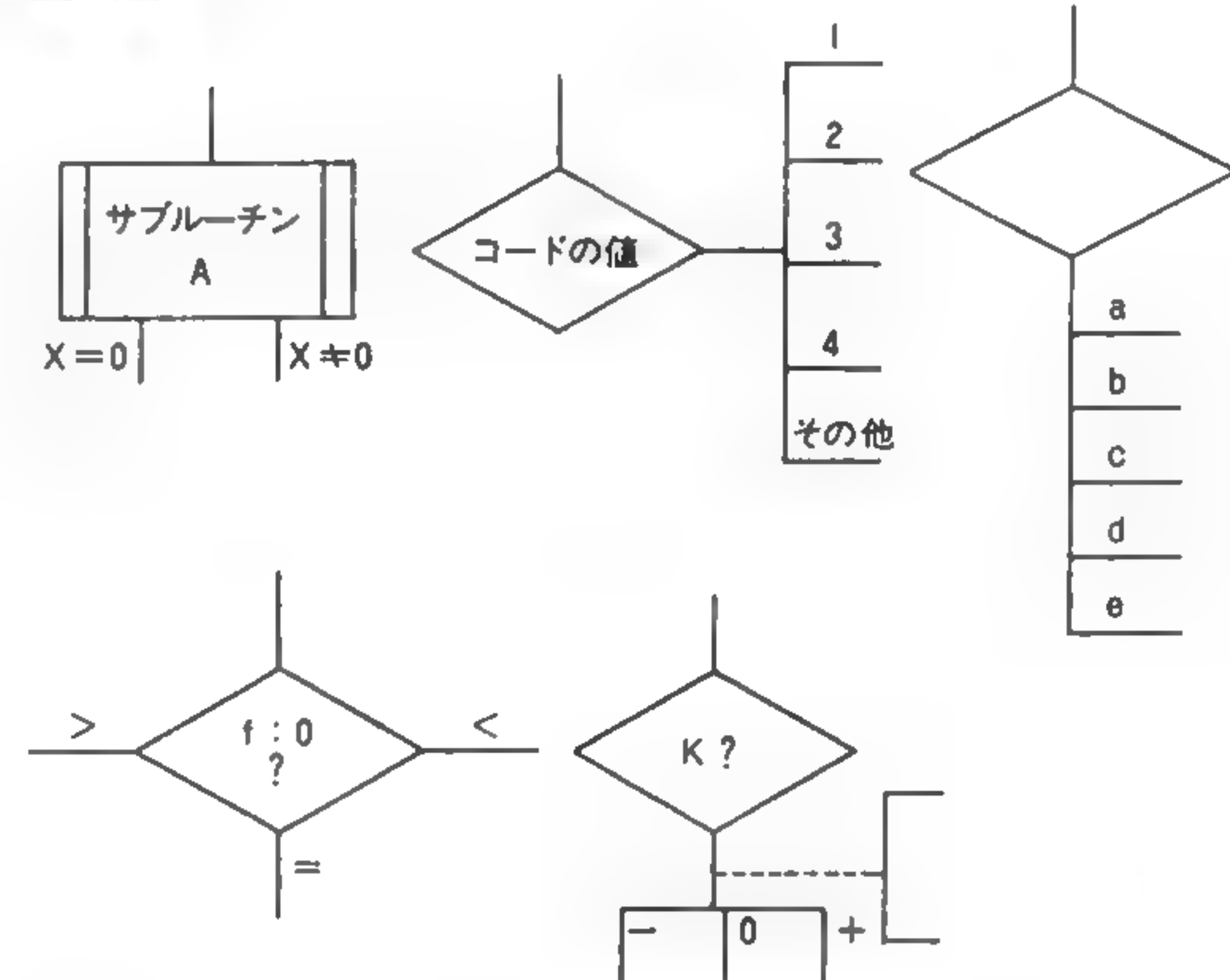
①分岐の表現 一つの流れ図記号からの出口を二つ以上書く場合には、必要な数だけの流れ線をその記号から出すか、又はその記号から出た流れ線が必要な数だけに分岐する形で表現する。

②分岐条件の記入 一つの流れ図記号に二つ以上の出口がある場合には、それぞれの出口に分岐条件を記入しなければならない。

〈備考〉

分岐条件を組にした決定表(decision table)を参照してもよい。

例：図- 7



8. 同種類の媒体の反復表現

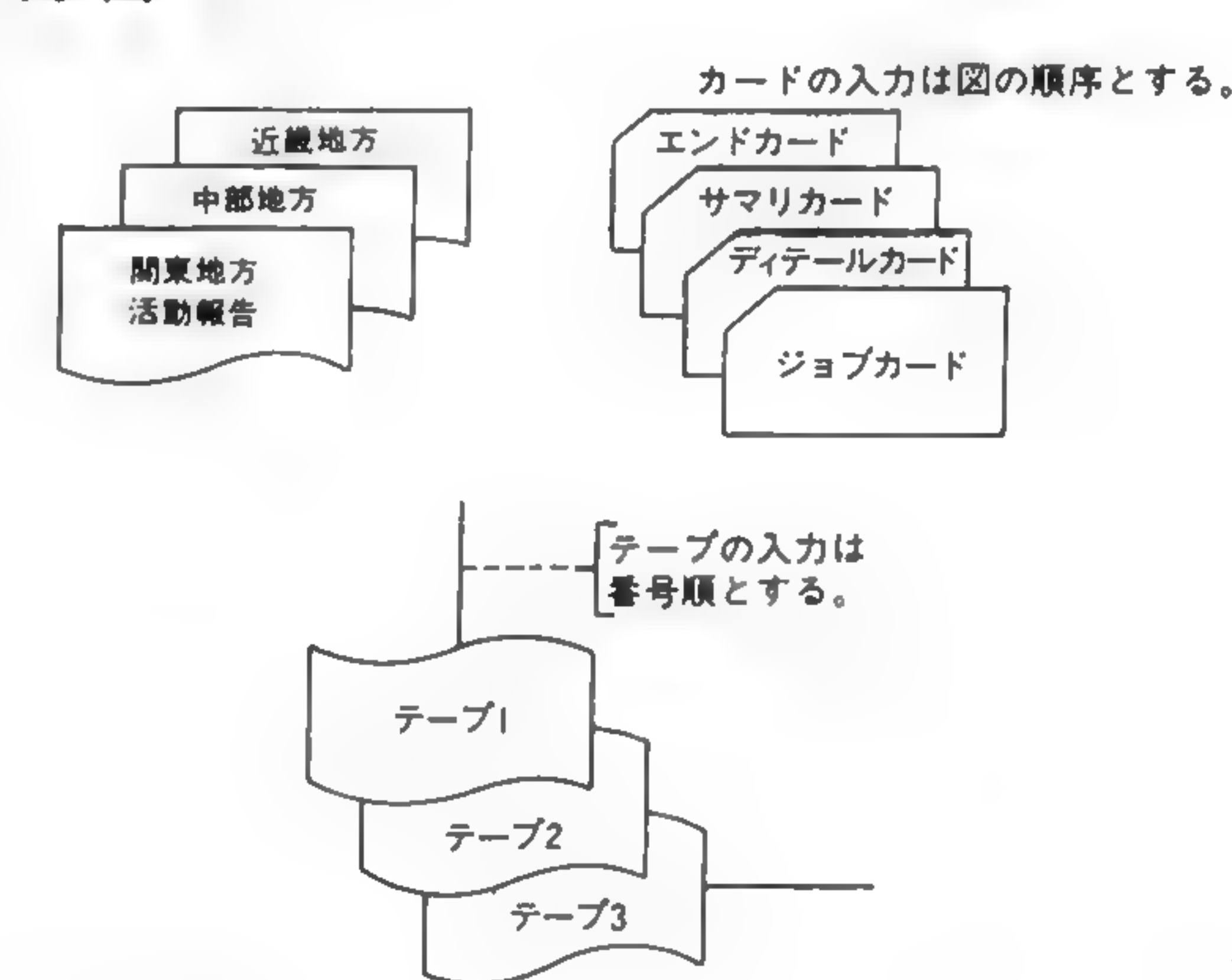
①反 復 多数の写し、各種の印刷形式の報告書、各種のせん孔形式の紙カードなど二つ以上の媒体やファイルを使用したり作成したりすることを示すには、必要な本文を記入した入出力用の単一の流れ図記号を用いる代わりに、同じ記号を重ねて書き表してもよい。

②位 置 重ねて書く場合には、先頭の流れ図記号は完全な記号の形式で書く。その後ろへ同じ記号を反復するには、先頭の記号から上又は下、かつ右又は左にずらして2番目の記号を書く。3番目以降の記号についても、同じ方向にずらしながら書かなければならない。

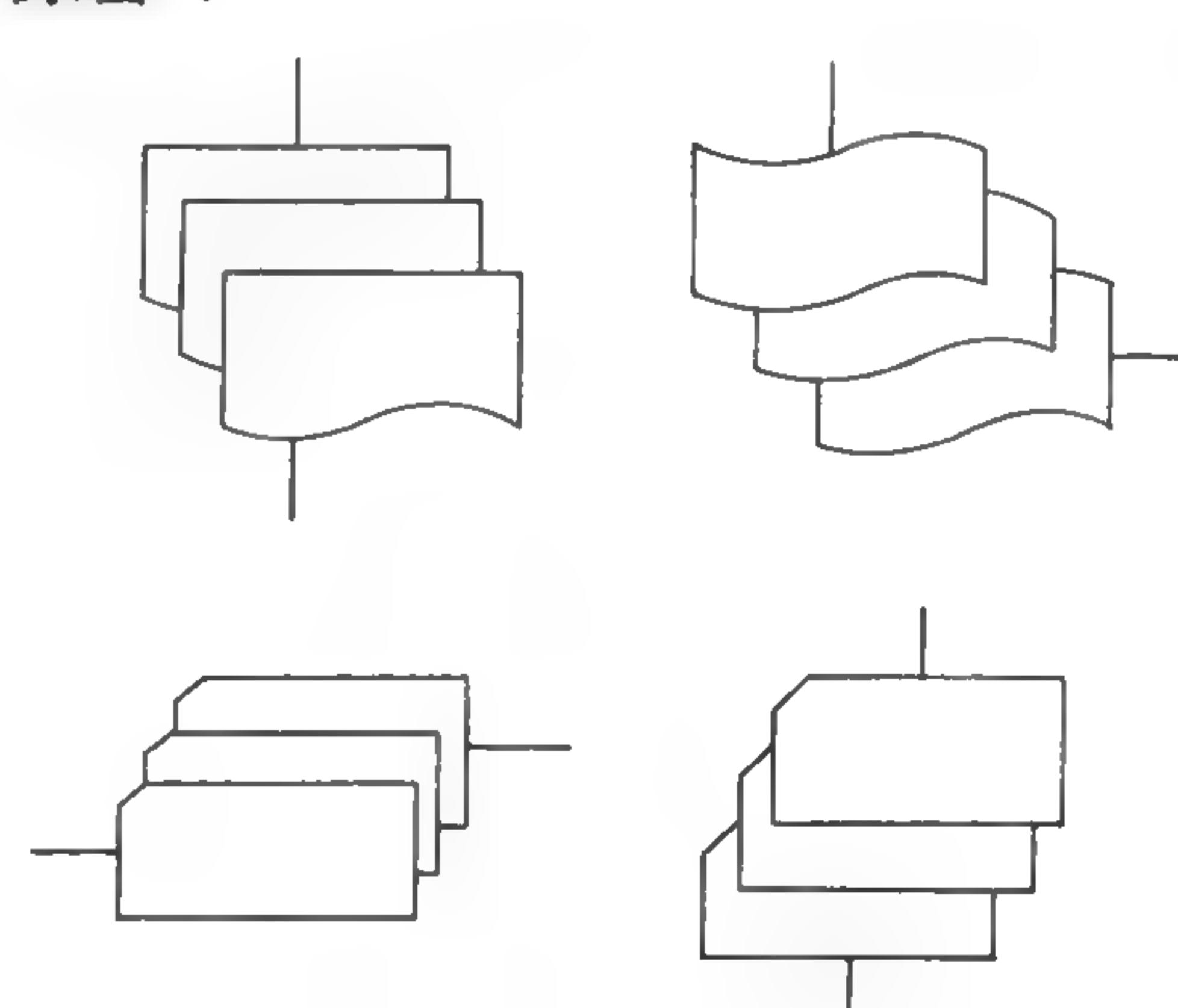
③順 序 重ねて書いた流れ図記号が順序の意味を含んでいる場合には、その順序は、前（先頭）から後ろ（末尾）に向かうものとし、かつそのことを明示する。

④流れ線 流れ線は、重ねて書いた流れ図記号のどの点に入り、また、どの点から出てもよい。重ねて書いた記号について8.③で定めた順序は、流れ線の出入りの点のいかんによっても変更されない。

例：図- 8



例：図- 9



MSXマシンのキャラクタ・コード表

		上 位 4 ビ ッ ト															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
下 位 4 ビ ッ ト	0		π		0	@	P		p	♠			ー	タ	ミ	た	み
	1	月		!	1	A	Q	a	q	♥	あ	。	ア	チ	ム	ち	む
	2	火		"	2	B	R	b	r	♣	い	「	イ	ツ	メ	つ	め
	3	水		#	3	C	S	c	s	♦	う	」	ウ	テ	モ	て	も
	4	木		\$	4	D	T	d	t	○	え	、	エ	ト	ヤ	と	や
	5	金		%	5	E	U	e	u	●	お	・	オ	ナ	ユ	な	ゆ
	6	土		&	6	F	V	f	v	を	か	ヲ	カ	ニ	ヨ	に	よ
	7	日		'	7	G	W	g	w	あ	き	ア	キ	ヌ	ラ	ぬ	ら
	8	年		(8	H	X	h	x	い	く	イ	ク	ネ	リ	ね	り
	9	円)	9	I	Y	i	y	う	け	ウ	ケ	ノ	ル	の	る
	A	時		*	:	J	Z	j	z	え	こ	エ	コ	ハ	レ	は	れ
	B	分		+	:	K	[k	{	お	さ	オ	サ	ヒ	ロ	ひ	ろ
	C	秒		,	<	L	¥	l		や	し	ヤ	シ	フ	ワ	ふ	わ
	D	百	大	ー	=	M]	m	!	ゆ	す	ユ	ス	ヘ	ン	へ	ん
	E	千	中	・	>	N	^	n	~	よ	せ	ヨ	セ	ホ	ゝ	ほ	
	F	万	小	/	?	O	—	o		っ	そ	ッ	ソ	マ	。	ま	

Z80Aインストラクション一覧表

1 8ビット・ロード命令 (フラグ表記) ・=影響受けない, 0=リセット, 1=セット, X=不定, ↑=演算結果に従った影響を受ける

ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン サイクル 数	ステート 数
		S	Z	H	P/V	N	C	76	543	210	16進			
LD r,s	$r \leftarrow s$	・	・	X	・	X	・	01	r	s		1	1	4
LD r,n	$r \leftarrow n$	・	・	X	・	X	・	00	r	110		2	2	7
								←	n	→				
LD r,(HL)	$r \leftarrow (HL)$	・	・	X	・	X	・	01	r	110		1	2	7
LD r,(IX+d)	$r \leftarrow (IX+d)$	・	・	X	・	X	・	11	011	101	DD	3	5	19
								01	r	110				
								←	d	→				
LD r,(IY+d)	$r \leftarrow (IY+d)$	・	・	X	・	X	・	11	111	101	FD	3	5	19
								01	r	110				
								←	d	→				
LD (HL),r	$(HL) \leftarrow r$	・	・	X	・	X	・	01	110	r		1	2	7
LD (IX+d),r	$(IX+d) \leftarrow r$	・	・	X	・	X	・	11	011	101	DD	3	5	19
								01	110	r				
								←	d	→				
LD (IY+d),r	$(IY+d) \leftarrow r$	・	・	X	・	X	・	11	111	101	FD	3	5	19
								01	110	r				
								←	d	→				
LD (HL),n	$(HL) \leftarrow n$	・	・	X	・	X	・	00	110	110	36	2	3	10
								←	n	→				
LD (IX+d),n	$(IX+d) \leftarrow n$	・	・	X	・	X	・	11	011	101	DD	4	5	19
								00	110	110	36			
								←	d	→				
								←	n	→				
LD (IY+d),n	$(IY+d) \leftarrow n$	・	・	X	・	X	・	11	111	101	FD	4	5	19
								00	110	110	36			
								←	d	→				
								←	n	→				
LD A,(BC)	$A \leftarrow (BC)$	・	・	X	・	X	・	00	001	010	0A	1	2	7
LD A,(DE)	$A \leftarrow (DE)$	・	・	X	・	X	・	00	011	010	1A	1	2	7
LD A,(nn)	$A \leftarrow (nn)$	・	・	X	・	X	・	00	111	010	3A	3	4	13
								←	n	→				
								←	n	→				
LD (BC),A	$(BC) \leftarrow A$	・	・	X	・	X	・	00	000	010	02	1	2	7
LD (DE),A	$(DE) \leftarrow A$	・	・	X	・	X	・	00	010	010	12	1	2	7
LD (nn),A	$(nn) \leftarrow A$	・	・	X	・	X	・	00	110	010	32	3	4	13
								←	n	→				
								←	n	→				
LD A,I	$A \leftarrow I$	↑	↑	X	0	X	IFF 0	11	101	101	ED	2	2	9
								01	010	111	57			
LD A,R	$A \leftarrow R$	↑	↑	X	0	X	IFF 0	11	101	101	ED	2	2	9
								01	011	111	5F			
LD I,A	$I \leftarrow A$	・	・	X	・	X	・	11	101	101	ED	2	2	9
								01	000	111	47			
LD R,A	$R \leftarrow A$	・	・	X	・	X	・	11	101	101	ED	2	2	9
								01	001	111	4F			

＜備考＞ r,sはレジスタA,B,C,D,E,H,Lを意味します。

r,s	レジスタ
000	B
001	C
010	D
011	E
100	H
101	L
111	A

IFFは割込み許可フリップ・フロップ(IFF)の内容です。

2 16ビット・ロード命令

ニモニク	動作	フ ラ グ						OPコード				バイト 数	マシン サイクル 数	ステート 数
		S	Z	H	P/V	N	C	76	543	210	16進			
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	•	•	3	3	10
								00	dd0	001				
								←	n	→				
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	•	DD	4	4	14
								11	011	101				
								00	100	001	21			
								←	n	→				
								←	n	→				
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	•	FD	4	4	14
								11	111	101				
								00	100	001	21			
								←	n	→				
								←	n	→				
LD HL, (nn)	H ← (nn + 1) L ← (nn)	•	•	X	•	X	•	•	•	•	2A	3	5	16
								00	101	010				
								←	n	→				
								←	n	→				
LD dd, (nn)	dd _H ← (nn + 1) dd _L ← (nn)	•	•	X	•	X	•	•	•	•	ED	4	6	20
								11	101	101				
								01	dd1	011				
								←	n	→				
								←	n	→				
LD IX, (nn)	IX _H ← (nn + 1) IX _L ← (nn)	•	•	X	•	X	•	•	•	•	DD	4	6	20
								11	011	101				
								00	101	010	2A			
								←	n	→				
								←	n	→				
LD IY, (nn)	IY _H ← (nn + 1) IY _L ← (nn)	•	•	X	•	X	•	•	•	•	FD	4	6	20
								11	111	101				
								00	101	010	2A			
								←	n	→				
								←	n	→				
LD (nn), HL	(nn + 1) ← H (nn) ← L	•	•	X	•	X	•	•	•	•	22	3	5	16
								00	100	010				
								←	n	→				
								←	n	→				
LD (nn), dd	(nn + 1) ← dd _H (nn) ← dd _L	•	•	X	•	X	•	•	•	•	ED	4	6	20
								11	101	101				
								01	dd0	011				
								←	n	→				
								←	n	→				
LD (nn), IX	(nn + 1) ← IX _H (nn) ← IX _L	•	•	X	•	X	•	•	•	•	DD	4	6	20
								11	011	101				
								00	100	010	22			
								←	n	→				
								←	n	→				
LD (nn), IY	(nn + 1) ← IY _H (nn) ← IY _L	•	•	X	•	X	•	•	•	•	FD	4	6	20
								11	111	101				
								00	100	010	22			
								←	n	→				
								←	n	→				
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	•	F9	1	1	6
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	•	DD	2	2	10
								11	011	101				
								11	111	001	F9			
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	•	FD	2	2	10
								11	111	101				
								11	111	001	F9			

〈備考〉 ddはペア・レジスタBC, DE, HL, SPを意味します。

dd	ペア
00	BC
01	DE
10	HL
11	SP

(ペア・レジスタ)_H, (ペア・レジスタ)_Lは各ペア・レジスタの上位または下位8ビットを意味します。

例: BC_L = C, AF_H = A

3 プッシュ、ポップ命令

ニモニク	動 作	フ ラ グ						OPコード				バイト 数	マシン・ サイクル 数	ステート 数		
		S	Z	H	P/V	N	C	76	543	210	16進					
PUSH qq	(SP-2)←qq _L (SP-1)←qq _H	•	•	X	•	X	•	•	•	11	qq0	101		1	3	11
PUSH IX	(SP-2)←IX _L (SP-1)←IX _H	•	•	X	•	X	•	•	•	11	011	101	DD	2	4	15
										11	100	101	E5			
PUSH IY	(SP-2)←IY _L (SP-1)←IY _H	•	•	X	•	X	•	•	•	11	111	101	FD	2	4	15
										11	100	101	E5			
POP qq	qq _H ←(SP+1) qq _L ←(SP)	•	•	X	•	X	•	•	•	11	qq0	001		1	3	10
POP IX	IX _H ←(SP+1) IX _L ←(SP)	•	•	X	•	X	•	•	•	11	011	101	DD	2	4	14
										11	100	001	E1			
POP IY	IY _H ←(SP+1) IY _L ←(SP)	•	•	X	•	X	•	•	•	11	111	101	FD	2	4	14
										11	100	001	E1			

〈備考〉 qqはペア・レジスタAF, BC, DE, HLを意味します。

qq	ペア
00	BC
01	DE
10	HL
11	AF

4 データ交換命令

ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン・ サイクル 数	ステート 数		
		S	Z	H	P/V	N	C	76	543	210	16進					
EX DE, HL	DE↔HL	•	•	X	•	X	•	•	•	11	101	011	EB	1	1	4
EX AF, AF'	AF↔AF'	•	•	X	•	X	•	•	•	00	001	000	08	1	1	4
EXX	(BC↔BC') (DE↔DE') (HL↔HL')	•	•	X	•	X	•	•	•	11	011	001	D9	1	1	4
EX (SP), HL	H↔(SP+1) L↔(SP)	•	•	X	•	X	•	•	•	11	100	011	E3	1	5	19
EX (SP), IX	IX _H ↔(SP+1)	•	•	X	•	X	•	•	•	11	011	101	DD	2	6	23
	IX _L ↔(SP)									11	100	011	E3			
EX (SP), IY	IY _H ↔(SP+1)	•	•	X	•	X	•	•	•	11	111	101	FD	2	6	23
	IY _L ↔(SP)									11	100	011	E3			

5 ブロック転送命令

ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン サイクル 数	ステート 数
		S	Z	H	P/V	N	C	76	543	210	16進			
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC-1	•	•	X	0	X	①	11 10	101 100	101 000	ED A0	2	4	16
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC-1 BC=0になるまで	•	•	X	0	X	0	11 10	101 110	101 000	ED B0	2	(BC≠0のとき) 5 4 (BC=0のとき)	21 16
LDD	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1	•	•	X	0	X	①	11 10	101 101	101 000	ED A8	2	4	16
LDDR	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 BC=0になるまで	•	•	X	0	X	0	11 10	101 111	101 000	ED B8	2	(BC≠0のとき) 5 4 (BC=0のとき)	21 16

6 ブロック・サーチ命令

ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン サイクル 数	ステート 数
		S	Z	H	P/V	N	C	76	543	210	16進			
CPI	A←(HL) HL←HL+1 BC←BC-1	②	①	↑	↑	X	↑	11 10	101 100	101 001	ED A1	2	4	16
CPIR	A←(HL) HL←HL+1 BC←BC-1 A=(HL)または BC=0まで	②	①	↑	↑	X	↑	11 10	101 110	101 001	ED B1	2	5 4	③ 21 16
CPD	A←(HL) HL←HL-1 BC←BC-1	②	①	↑	↑	X	↑	11 10	101 101	101 001	ED A9	2	4	16
CPDR	A←(HL) HL←HL-1 BC←BC-1 A=(HL)または BC=0まで	②	①	↑	↑	X	↑	11 10	101 111	101 001	ED B9	2	5 4	③ 21 16

<備考> ①もしBC-1=0ならばP/V=0, その他P/V=1 ④BC=0またはA=(HL)のとき。
 ②もしA=(HL)ならばZ=1, その他Z=0
 ③BC≠0かつA≠(HL)のとき

7 8ビット演算命令

ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン・ サイクル 数	ステート 数		
		S	Z	H	P _V	N	C	76	543	210	16進					
ADD A, r	A←A+r	↑	↑	X	↑	X	V	0	↑	10	000	r		1	1	4
ADD A, n	A←A+n	↑	↑	X	↑	X	V	0	↑	11	000	110		2	2	7
								←	n	→						
ADD A, (HL)	A←A+(HL)	↑	↑	X	↑	X	V	0	↑	10	000	110		1	2	7
ADD A, (IX+d)	A←A+(IX+d)	↑	↑	X	↑	X	V	0	↑	11	011	101	DD	3	5	19
								10	000	110						
								←	d	→						
ADD A, (IY+d)	A←A+(IY+d)	↑	↑	X	↑	X	V	0	↑	11	111	101	FD	3	5	19
								10	000	110						
								←	d	→						
ADC A, s	A←A+s+CY	↑	↑	X	↑	X	V	0	↑		001					
SUB s	A←A-s	↑	↑	X	↑	X	V	1	↑		010					
SBC A, s	A←A-s-CY	↑	↑	X	↑	X	V	1	↑		011					
AND s	A←A∧s	↑	↑	X	1	X	P	0	0		100		①			
OR s	A←A∨s	↑	↑	X	0	X	P	0	0		110					
XOR s	A←A⊕s	↑	↑	X	0	X	P	0	0		101					
CP s	A-s	↑	↑	X	↑	X	V	1	↑		111					
INC r	r←r+1	↑	↑	X	↑	X	V	0	•	00	r	100		1	1	4
INC (HL)	(HL)←(HL)+1	↑	↑	X	↑	X	V	0	•	00	110	100		1	3	11
INC (IX+d)	(IX+d)←(IX+d)+1	↑	↑	X	↑	X	V	0	•	11	011	101	DD	3	6	23
								00	110	100						
								←	d	→						
INC (IY+d)	(IY+d)←(IY+d)+1	↑	↑	X	↑	X	V	0	•	11	111	101	FD	3	6	23
								00	110	100						
								←	d	→						
DEC s	s←s-1	↑	↑	X	↑	X	V	1	•		101					
DAA	10進演算補正	↑	↑	X	↑	X	P	•	↑	00	100	111	27	1	1	4
CPL	A←A [¯]	•	•	X	1	X	•	1	•	00	101	111	2F	1	1	4
	(1の補数)															
NEG	A←A+1	↑	↑	X	↑	X	V	1	↑	11	101	101	ED	2	2	8
	(2の補数)							01	000	100	44					

＜備考＞ s=r, n, (HL), (IX+d), (IY+d)のいずれか。

① 000=001~111に変わるほかは ADD命令と同様な繰り返し。

INC命令→DEC命令=100→101に変わるほかは同じ。

r	レジスタ
000	B
001	C
010	D
011	E
100	H
101	L
111	A

P_Vフラグ

論理演算の場合：

偶数パリティ→P=1

奇数パリティ→P=0

算術演算の場合：

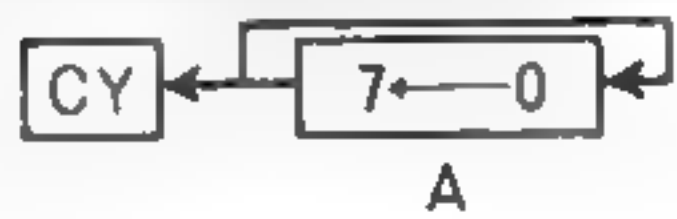
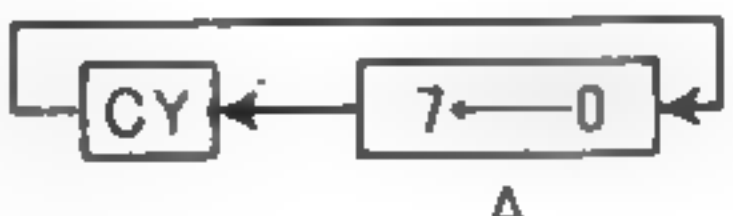

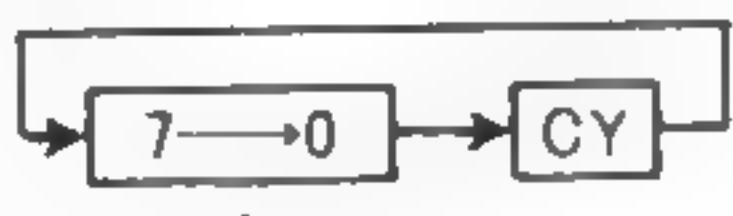
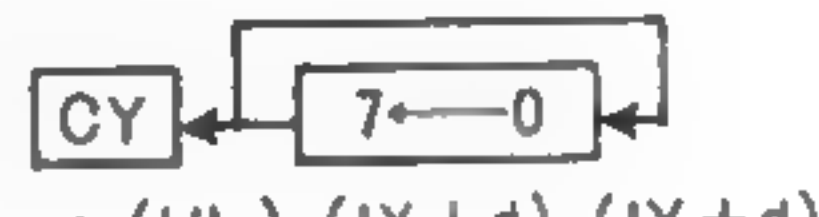
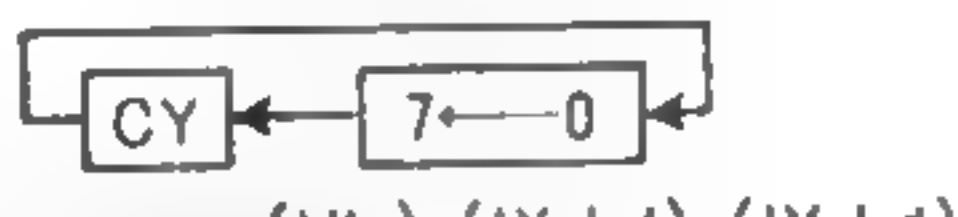

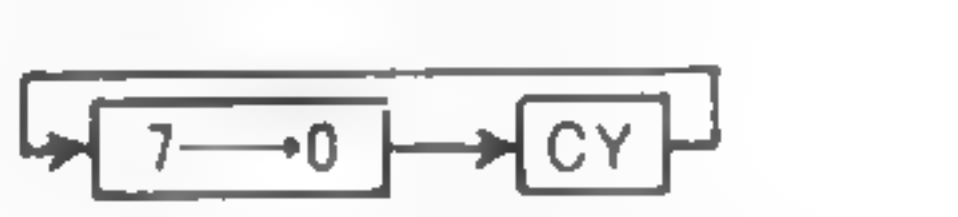
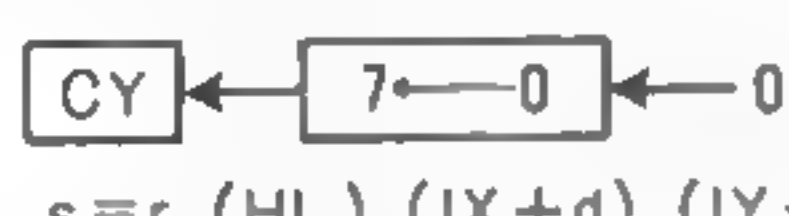
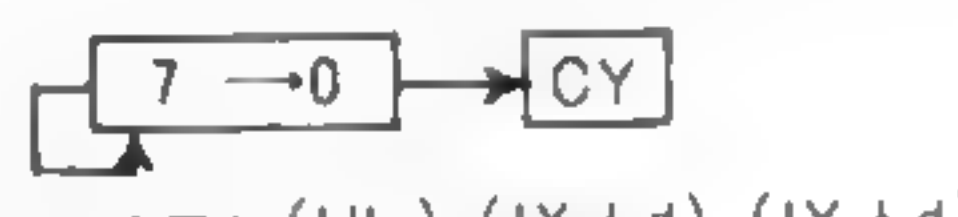
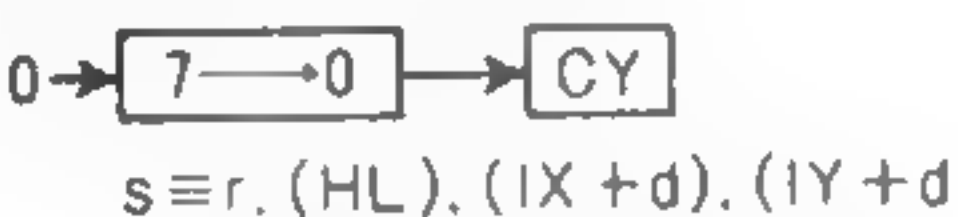

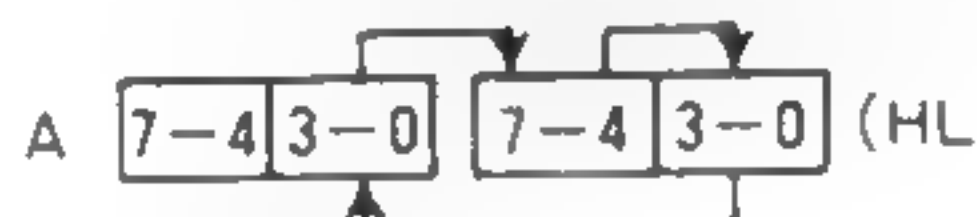
オーバーフローあり→V=1

オーバーフローなし→V=0

8 16ビット演算命令

ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン・ サイクル 数	ステート 数																																
		S	Z	H	P/V	N	C	76	543	210	16進																																			
ADD HL,ss	HL←HL+ss	•	•	X	X	X	•	0	∴	00	ss1	001		1	3	11																														
ADC HL,ss	HL←HL+ss+CY	∴	∴	X	X	X	V	0	∴	11	101	101	ED	2	4	15																														
								01	ss1	010																																				
SBC HL,ss	HL←HL−ss−CY	∴	∴	X	X	X	V	1	∴	11	101	101	ED	2	4	15																														
								01	ss0	010																																				
ADD IX,pp	IX←IX+pp	•	•	X	X	X	•	0	∴	11	011	101	DD	2	4	15																														
								00	pp1	001																																				
ADD IY,rr	IY←IY+rr	•	•	X	X	X	•	0	∴	11	111	101	FD	2	4	15																														
								00	rr1	001																																				
INC ss	ss←ss+1	•	•	X	•	X	•	•	•	00	ss0	011		1	1	6																														
INC IX	IX←IX+1	•	•	X	•	X	•	•	•	11	011	101	DD	2	2	10																														
								00	100	011	23																																			
INC IY	IY←IY+1	•	•	X	•	X	•	•	•	11	111	101	FD	2	2	10																														
								00	100	011	23																																			
DEC ss	ss←ss−1	•	•	X	•	X	•	•	•	00	ss1	011		1	1	6																														
DEC IX	IX←IX−1	•	•	X	•	X	•	•	•	11	011	101	DD	2	2	10																														
								00	101	011	2B																																			
DEC IY	IY←IY−1	•	•	X	•	X	•	•	•	11	111	101	FD	2	2	10																														
								00	101	011	2B																																			
<div>＜備考＞<table><tr><td>ss</td><td>レジスタ</td><td>pp</td><td>レジスタ</td><td>rr</td><td>レジスタ</td></tr><tr><td>00</td><td>BC</td><td>00</td><td>BC</td><td>00</td><td>BC</td></tr><tr><td>01</td><td>DE</td><td>01</td><td>DE</td><td>01</td><td>DE</td></tr><tr><td>10</td><td>HL</td><td>10</td><td>IX</td><td>10</td><td>IY</td></tr><tr><td>11</td><td>SP</td><td>11</td><td>SP</td><td>11</td><td>SP</td></tr></table></div>																	ss	レジスタ	pp	レジスタ	rr	レジスタ	00	BC	00	BC	00	BC	01	DE	01	DE	01	DE	10	HL	10	IX	10	IY	11	SP	11	SP	11	SP
ss	レジスタ	pp	レジスタ	rr	レジスタ																																									
00	BC	00	BC	00	BC																																									
01	DE	01	DE	01	DE																																									
10	HL	10	IX	10	IY																																									
11	SP	11	SP	11	SP																																									

9 ローテイト、シフト命令

ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン サイクル 数	ステート 数						
		S	Z	H	P/V	N	C	76	543	210	16進									
RLCA		•	•	X	0	X	•	0	↓	00	000	111	07	1	1	4				
RLA		•	•	X	0	X	•	0	↓	00	010	111	17	1	1	4				
RRCA		•	•	X	0	X	•	0	↓	00	001	111	0F	1	1	4				
RRA		•	•	X	0	X	•	0	↓	00	011	111	1F	1	1	4				
RLC r		↑	↑	X	0	X	P	0	↓	11	001	011	CB	2	2	8				
RLC (HL)		↑	↑	X	0	X	P	0	↓	11	001	011	CB	2	4	15				
RLC (IX+d)		↑	↑	X	0	X	P	0	↓	11	011	101	DD	4	6	23				
RLC (IY+d)		↑	↑	X	0	X	P	0	↓	11	001	011	CB	4	6	23				
		←						d	→	00	000	110								
		↑	↑	X	0	X	P	0	↓	11	111	101	FD							
		←						d	→	00	001	011	CB							
RL s		↑	↑	X	0	X	P	0	↓		010			①						
RRC s		↑	↑	X	0	X	P	0	↓		001									
RR s		↑	↑	X	0	X	P	0	↓		011									
SLA s		↑	↑	X	0	X	P	0	↓		100									
SRA s		↑	↑	X	0	X	P	0	↓		101									
SRL s		↑	↑	X	0	X	P	0	↓		111									
RLD		↑	↑	X	0	X	P	0	•	11	101	101	ED	2	5	18				
RRD		↑	↑	X	0	X	P	0	•	01	101	111	6F	2	5	18				
		↑	↑	X	0	X	P	0	•	11	101	101	ED							
＜備考＞		r	① 000 = 001 ~ 111 に変わるほかは、RLC命令と同様な繰り返し。																	
		000	B																	
		001	C																	
		010	D																	
		011	E																	
		100	H																	
		101	L																	
		111	A																	

10 ビット操作命令

ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン・ サイクル 数	ステート 数		
		S	Z	H	P/V	N	C	76	543	210	16進					
BIT b,r	$Z \leftarrow \bar{r}_b$	x	;	x	1	x	x	0	•	11	001	011	CB	2	2	8
								01		b		r				
BIT b,(HL)	$Z \leftarrow \overline{(HL)_b}$	x	;	x	1	x	x	0	•	11	001	011	CB	2	3	12
								01		b		110				
BIT b,(IX+d)	$Z \leftarrow \overline{(IX+d)_b}$	x	;	x	1	x	x	0	•	11	011	101	DD	4	5	20
								11		001		011	CB			
								←		d		→				
								01		b		110				
BIT b,(IY+d)	$Z \leftarrow \overline{(IY+d)_b}$	x	;	x	1	x	x	0	•	11	111	101	FD	4	5	20
								11		001		011	CB			
								←		d		→				
								01		b		110				
SET b,r	$r_b \leftarrow 1$	•	•	x	•	x	•	•	•	11	001	011	CB	2	2	8
								11		b		r				
SET b,(HL)	$(HL)_b \leftarrow 1$	•	•	x	•	x	•	•	•	11	001	011	CB	2	4	15
								11		b		110				
SET b,(IX+d)	$(IX+d)_b \leftarrow 1$	•	•	x	•	x	•	•	•	11	011	101	DD	4	6	23
								11		001		011	CB			
								←		d		→				
								11		b		110				
SET b,(IY+d)	$(IY+d)_b \leftarrow 1$	•	•	x	•	x	•	•	•	11	111	101	FD	4	6	23
								11		001		011	CB			
								←		d		→				
								11		b		110				
RES b,s	$s_b \leftarrow 0$ $s \equiv r, (HL), (IX+d), (IY+d)$							10								

<備考>

r	レジスタ	b	Bit
000	B	000	0
001	C	001	1
010	D	010	2
011	E	011	3
100	H	100	4
101	L	101	5
110		110	6
111	A	111	7

SET命令→RES命令=11→10に変えて同様な繰り返し

11 フラグ操作命令

ニモニク	動 作	フ ラ グ						OPコード				バイト 数	マシン・ サイクル 数	ステート 数		
		S	Z	H	P/V	N	C	76	543	210	16進					
CCF	CY・ $\overline{\text{CY}}$	・	・	×	×	×	・	0	:	00	111	111	3F	1	1	4
SCF	CY・ 1	・	・	×	0	×	・	0	1	00	110	111	37	1	1	4

12 ジャンプ命令

ニモニク	動 作	フ ラ グ						OPコード				バイト 数	マシン サイクル 数	ステート 数
		S	Z	H	P/V	N	C	76	543	210	16進			
JP nn	PC←nn	•	•	x	•	x	•	•	•	•	C3	3	3	10
								←	n	→				
								←	n	→				
JP cc, nn	もしccが成り立てばジャンプ。そうでなければ次命令へ。	•	•	x	•	x	•	•	•	•		3	3	10
								←	n	→				
								←	n	→				
JR e	PC←PC+e	•	•	x	•	x	•	•	•	•	18	2	3	12
								←	e-2	→				
JR C, e	C=0なら次命令へ。	•	•	x	•	x	•	•	•	•	38	2	2	7
	C=1なら							←	e-2	→		2	3	12
	PC←PC+e													
JR NC, e	C=1なら次命令へ。	•	•	x	•	x	•	•	•	•	30	2	2	7
	C=0なら							←	e-2	→		2	3	12
	PC←PC+e													
JR Z, e	Z=0なら次命令へ。	•	•	x	•	x	•	•	•	•	28	2	2	7
	Z=1なら							←	e-2	→		2	3	12
	PC←PC+e													
JR NZ, e	Z=1なら次命令へ。	•	•	x	•	x	•	•	•	•	20	2	2	7
	Z=0なら							←	e-2	→		2	3	12
	PC←PC+e													
JP (HL)	PC←HL	•	•	x	•	x	•	•	•	•	E9	1	1	4
JP (IX)	PC←IX	•	•	x	•	x	•	•	•	•	DD	2	2	8
								11	101	001	E9			
JP (IY)	PC←IY	•	•	x	•	x	•	•	•	•	FD	2	2	8
								11	101	001	E9			
DJNZ e	B←B-1	•	•	x	•	x	•	•	•	•	10	2	2	8
	B=0なら次命令へ。							←	e-2	→		2	3	13
	B≠0ならPC←PC+e													
〈備考〉 e=レラティブ・アトレッシング・モードにおける変位値 (符号付き2の補数=+127〜-128) e-2=eの実効変位値(レラティブ・アトレッシングの項参照)									cc	条件				
									000	NZ				
									001	Z				
									010	NC				
									011	C				
									100	PO				
									101	PE				
									110	P				
									111	M				

13 コール, リターン命令

ニモニツク	動 作	フ ラ グ						OPコード				バイト 数	マシン サイクル 数	ステート 数				
		S	Z	H	P _V	N	C	76	543	210	16進							
CALL nn	(SP-1)←PC _H	•	•	X	•	X	•	•	•	•	11	001	101	CD	3	5	17	
	(SP-2)←PC _L										←	n	→					
	PC←nn											←	n	→				
CALL cc,nn	もしccが成り立てばコール。そうでなければ次命令へ。	•	•	X	•	X	•	•	•	•	11	cc	100		3	3	10	
												←	n	→				
												←	n	→		3	5	17
RET	PC _L ←(SP) PC _H ←(SP+1)	•	•	X	•	X	•	•	•	•	11	001	001	C9	1	3	10	
RET cc	もしccが成り立てばリターン。そうでなければ次命令へ。	•	•	X	•	X	•	•	•	•	11	cc	000		1	1	5	
																1	3	11
RETI	インタラプトから戻る。	•	•	X	•	X	•	•	•	•	11	101	101	ED	2	4	14	
RETN	ノンマスカブル・インタラプトから戻る。										01	001	101	4D				
		•	•	X	•	X	•	•	•	•	11	101	101	ED	2	4	14	
											01	000	101	45				
RST p	(SP-1)←PC _H (SP-2)←PC _L PC _H ←0 PC _L ←p	•	•	X	•	X	•	•	•	•	11	t	111		1	3	11	

cc	条件	t	p
000	NZ	000	00H
001	Z	001	08H
010	NC	010	10H
011	C	011	18H
100	PO	100	20H
101	PE	101	28H
110	P	110	30H
111	M	111	38H

14 入出力命令

ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン サイクル 数	ステート 数
		S	Z	H	P/V	N	C	76	543	210	16進			
IN A, (n)	$A \leftarrow (n)$	•	•	X	•	X	•	•	•	•	DB	2	3	11
IN r, (C)	$r \leftarrow (C)$	↑	↑	X	0	X	P	0	•	← n →	ED	2	3	12
INI	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$	①	X	↑	X	X	X	X	I	•	ED	2	4	16
INIR	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$ $B = 0$ になるまでくりかえす	X	I	X	X	X	X	I	•	ED	2	5	21	(If B≠0)
IND	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$	①	X	↑	X	X	X	X	I	•	ED	2	4	16
INDR	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$ $B = 0$ になるまでくりかえす	X	I	X	X	X	X	I	•	ED	2	5	21	(If B≠0)
OUT (n).A	$(n) \leftarrow A$	•	•	X	•	X	•	•	•	← n →	D3	2	3	11
OUT (C).r	$(C) \leftarrow r$	•	•	X	•	X	•	•	•	← r →	ED	2	3	12
OUTI	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$	①	X	↑	X	X	X	X	I	•	ED	2	4	16
OTIR	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$ $B = 0$ になるまでくりかえす	X	I	X	X	X	X	I	•	ED	2	5	21	(If B≠0)
OUTD	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$	①	X	↑	X	X	X	X	I	•	ED	2	4	16
OTDR	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$ $B = 0$ になるまでくりかえす	X	I	X	X	X	X	I	•	ED	2	5	21	(If B≠0)
		X	I	X	X	X	X	I	•	ED	2	4	16	(If B=0)

〈備考〉 ①もし $B - 1 = 0$ ならば $Z = 1$, その他 $Z = 0$ r は 10.ビット操作命令の備考を参照

15 CPUコントロール命令


ニモニック	動 作	フ ラ グ						OPコード				バイト 数	マシン・ サイクル 数	ステート 数	
		S	Z	H	P/V	N	C	76	543	210	16進				
NOP	No operation	・	・	X	・	X	・	・	00	000	000	00	1	1	4
HALT	CPU停止	・	・	X	・	X	・	・	01	110	110	76	1	1	4
DI	IFF←0	・	・	X	・	X	・	・	11	110	011	F3	1	1	4
EI	IFF←1	・	・	X	・	X	・	・	11	111	011	FB	1	1	4
IM 0	割込モード0	・	・	X	・	X	・	・	11	101	101	ED	2	2	8
									01	000	110	46			
IM 1	割込モード1	・	・	X	・	X	・	・	11	101	101	ED	2	2	8
									01	010	110	56			
IM 2	割込モード2	・	・	X	・	X	・	・	11	101	101	ED	2	2	8
									01	011	110	5E			
〈備考〉 IFF=割込許可フリップ・フロップ															

Z-80A機械語↔ニモニック対応表

機 械 語 ← → ニ モ ニ ッ ク											
00	NOP		40	LD	B, B	80	ADD	A, B	C0	RET	NZ
01	LD	BC, nn	41	LD	B, C	81	ADD	A, C	C1	POP	BC
02	LD	(BC), A	42	LD	B, D	82	ADD	A, D	C2	JP	NZ, nn
03	INC	BC	43	LD	B, E	83	ADD	A, E	C3	JP	nn
04	INC	B	44	LD	B, H	84	ADD	A, H	C4	CALL	NZ, nn
05	DEC	B	45	LD	B, L	85	ADD	A, L	C5	PUSH	BC
06	LD	B, n	46	LD	B, (HL)	86	ADD	A, (HL)	C6	ADD	A, n
07	RLCA		47	LD	B, A	87	ADD	A, A	C7	RST	00H
08	EX	AF, AF'	48	LD	C, B	88	ADC	A, B	C8	RET	Z
09	ADD	HL, BC	49	LD	C, C	89	ADC	A, C	C9	RET	
0A	LD	A, (BC)	4A	LD	C, D	8A	ADC	A, D	CA	JP	Z, nn
0B	DEC	BC	4B	LD	C, E	8B	ADC	A, E	CB		
0C	INC	C	4C	LD	C, H	8C	ADC	A, H	CC	CALL	Z, nn
0D	DEC	C	4D	LD	C, L	8D	ADC	A, L	CD	CALL	nn
0E	LD	C, n	4E	LD	C, (HL)	8E	ADC	A, (HL)	CE	ADC	A, n
0F	RRCA		4F	LD	C, A	8F	ADC	A, A	CF	RST	08H
10	DJNZ	e	50	LD	D, B	90	SUB	B	D0	RET	NC
11	LD	DE, nn	51	LD	D, C	91	SUB	C	D1	POP	DE
12	LD	(DE), A	52	LD	D, D	92	SUB	D	D2	JP	NC, nn
13	INC	DE	53	LD	D, E	93	SUB	E	D3	OUT	n, A
14	INC	D	54	LD	D, H	94	SUB	H	D4	CALL	NC, nn
15	DEC	D	55	LD	D, L	95	SUB	L	D5	PUSH	DE
16	LD	D, n	56	LD	D, (HL)	96	SUB	(HL)	D6	SUB	n
17	RLA		57	LD	D, A	97	SUB	A	D7	RST	10H
18	JR	e	58	LD	E, B	98	SBC	A, B	D8	RET	C
19	ADD	HL, DE	59	LD	E, C	99	SBC	A, C	D9	EXX	
1A	LD	A, (DE)	5A	LD	E, D	9A	SBC	A, D	DA	JP	C, nn
1B	DEC	DE	5B	LD	E, E	9B	SBC	A, E	DB	IN	A, n
1C	INC	E	5C	LD	E, H	9C	SBC	A, H	DC	CALL	C, nn
1D	DEC	E	5D	LD	E, L	9D	SBC	A, L	DD		
1E	LD	E, n	5E	LD	E, (HL)	9E	SBC	A, (HL)	DE	SBC	A, n
1F	RRA		5F	LD	E, A	9F	SBC	A, A	DF	RST	18H
20	JR	NZ, e	60	LD	H, B	A0	AND	B	E0	RET	PO
21	LD	HL, nn	61	LD	H, C	A1	AND	C	E1	POP	HL
22	LD	(nn), HL	62	LD	H, D	A2	AND	D	E2	JP	PO, nn
23	INC	HL	63	LD	H, E	A3	AND	E	E3	EX	(SP), HL
24	INC	H	64	LD	H, H	A4	AND	H	E4	CALL	PO, nn
25	DEC	H	65	LD	H, L	A5	AND	L	E5	PUSH	HL
26	LD	H, n	66	LD	H, (HL)	A6	AND	(HL)	E6	AND	n
27	DAA		67	LD	H, A	A7	AND	A	E7	RST	20H
28	JR	Z, e	68	LD	L, B	A8	XOR	B	E8	RET	PE
29	ADD	HL, HL	69	LD	L, C	A9	XOR	C	E9	JP	(HL)
2A	LD	HL, (nn)	6A	LD	L, D	AA	XOR	D	EA	JP	PE, nn
2B	DEC	HL	6B	LD	L, E	AB	XOR	E	EB	EX	DE, HL
2C	INC	L	6C	LD	L, H	AC	XOR	H	EC	CALL	PE, nn
2D	DEC	L	6D	LD	L, L	AD	XOR	L	ED		
2E	LD	L, n	6E	LD	L, (HL)	AE	XOR	(HL)	EE	XOR	n
2F	CPL		6F	LD	L, A	AF	XOR	A	EF	RST	28H
30	JR	NC, e	70	LD	(HL), B	B0	OR	B	F0	RET	P
31	LD	SP, nn	71	LD	(HL), C	B1	OR	C	F1	POP	AF
32	LD	(nn), A	72	LD	(HL), D	B2	OR	D	F2	JP	P, nn
33	INC	SP	73	LD	(HL), E	B3	OR	E	F3	DI	
34	INC	(HL)	74	LD	(HL), H	B4	OR	H	F4	CALL	P, nn
35	DEC	(HL)	75	LD	(HL), L	B5	OR	L	F5	PUSH	AF
36	LD	(HL), n	76	HALT		B6	OR	(HL)	F6	OR	n
37	SCF		77	LD	(HL), A	B7	OR	A	F7	RST	30H
38	JR	C, e	78	LD	A, B	B8	CP	B	F8	RET	M
39	ADD	HL, SP	79	LD	A, C	B9	CP	C	F9	LD	SP, HL
3A	LD	A, (nn)	7A	LD	A, D	BA	CP	D	FA	JP	M, nn
3B	DEC	SP	7B	LD	A, E	BB	CP	E	FB	EI	
3C	INC	A	7C	LD	A, H	BC	CP	H	FC	CALL	M, nn
3D	DEC	A	7D	LD	A, L	BD	CP	L	FD		
3E	LD	A, n	7E	LD	A, (HL)	BE	CP	(HL)	FE	CP	n
3F	CCF		7F	LD	A, A	BF	CP	A	FF	RST	38H

CB XX															
00	RLC	B		40	BIT	0, B		80	RES	0, B		C0	SET	0, B	
01	RLC	C		41	BIT	0, C		81	RES	0, C		C1	SET	0, C	
02	RLC	D		42	BIT	0, D		82	RES	0, D		C2	SET	0, D	
03	RLC	E		43	BIT	0, E		83	RES	0, E		C3	SET	0, E	
04	RLC	H		44	BIT	0, H		84	RES	0, H		C4	SET	0, H	
05	RLC	L		45	BIT	0, L		85	RES	0, L		C5	SET	0, L	
06	RLC	(HL)		46	BIT	0, (HL)		86	RES	0, (HL)		C6	SET	0, (HL)	
07	RLC	A		47	BIT	0, A		87	RES	0, A		C7	SET	0, A	
08	RRC	B		48	BIT	1, B		88	RES	1, B		C8	SET	1, B	
09	RRC	C		49	BIT	1, C		89	RES	1, C		C9	SET	1, C	
0A	RRC	D		4A	BIT	1, D		8A	RES	1, D		CA	SET	1, D	
0B	RRC	E		4B	BIT	1, E		8B	RES	1, E		CB	SET	1, E	
0C	RRC	H		4C	BIT	1, H		8C	RES	1, H		CC	SET	1, H	
0D	RRC	L		4D	BIT	1, L		8D	RES	1, L		CD	SET	1, L	
0E	RRC	(HL)		4E	BIT	1, (HL)		8E	RES	1, (HL)		CE	SET	1, (HL)	
0F	RRC	A		4F	BIT	1, A		8F	RES	1, A		CF	SET	1, A	
10	RL	B		50	BIT	2, B		90	RES	2, B		D0	SET	2, B	
11	RL	C		51	BIT	2, C		91	RES	2, C		D1	SET	2, C	
12	RL	D		52	BIT	2, D		92	RES	2, D		D2	SET	2, D	
13	RL	E		53	BIT	2, E		93	RES	2, E		D3	SET	2, E	
14	RL	H		54	BIT	2, H		94	RES	2, H		D4	SET	2, H	
15	RL	L		55	BIT	2, L		95	RES	2, L		D5	SET	2, L	
16	RL	(HL)		56	BIT	2, (HL)		96	RES	2, (HL)		D6	SET	2, (HL)	
17	RL	A		57	BIT	2, A		97	RES	2, A		D7	SET	2, A	
18	RR	B		58	BIT	3, B		98	RES	3, B		D8	SET	3, B	
19	RR	C		59	BIT	3, C		99	RES	3, C		D9	SET	3, C	
1A	RR	D		5A	BIT	3, D		9A	RES	3, D		DA	SET	3, D	
1B	RR	E		5B	BIT	3, E		9B	RES	3, E		DB	SET	3, E	
1C	RR	H		5C	BIT	3, H		9C	RES	3, H		DC	SET	3, H	
1D	RR	L		5D	BIT	3, L		9D	RES	3, L		DD	SET	3, L	
1E	RR	(HL)		5E	BIT	3, (HL)		9E	RES	3, (HL)		DE	SET	3, (HL)	
1F	RR	A		5F	BIT	3, A		9F	RES	3, A		DF	SET	3, A	
20	SLA	B		60	BIT	4, B		A0	RES	4, B		E0	SET	4, B	
21	SLA	C		61	BIT	4, C		A1	RES	4, C		E1	SET	4, C	
22	SLA	D		62	BIT	4, D		A2	RES	4, D		E2	SET	4, D	
23	SLA	E		63	BIT	4, E		A3	RES	4, E		E3	SET	4, E	
24	SLA	H		64	BIT	4, H		A4	RES	4, H		E4	SET	4, H	
25	SLA	L		65	BIT	4, L		A5	RES	4, L		E5	SET	4, L	
26	SLA	(HL)		66	BIT	4, (HL)		A6	RES	4, (HL)		E6	SET	4, (HL)	
27	SLA	A		67	BIT	4, A		A7	RES	4, A		E7	SET	4, A	
28	SRA	B		68	BIT	5, B		A8	RES	5, B		E8	SET	5, B	
29	SRA	C		69	BIT	5, C		A9	RES	5, C		E9	SET	5, C	
2A	SRA	D		6A	BIT	5, D		AA	RES	5, D		EA	SET	5, D	
2B	SRA	E		6B	BIT	5, E		AB	RES	5, E		EB	SET	5, E	
2C	SRA	H		6C	BIT	5, H		AC	RES	5, H		EC	SET	5, H	
2D	SRA	L		6D	BIT	5, L		AD	RES	5, L		ED	SET	5, L	
2E	SRA	(HL)		6E	BIT	5, (HL)		AE	RES	5, (HL)		EE	SET	5, (HL)	
2F	SRA	A		6F	BIT	5, A		AF	RES	5, A		EF	SET	5, A	
30				70	BIT	6, B		B0	RES	6, B		F0	SET	6, B	
31				71	BIT	6, C		B1	RES	6, C		F1	SET	6, C	
32				72	BIT	6, D		B2	RES	6, D		F2	SET	6, D	
33				73	BIT	6, E		B3	RES	6, E		F3	SET	6, E	
34				74	BIT	6, H		B4	RES	6, H		F4	SET	6, H	
35				75	BIT	6, L		B5	RES	6, L		F5	SET	6, L	
36				76	BIT	6, (HL)		B6	RES	6, (HL)		F6	SET	6, (HL)	
37				77	BIT	6, A		B7	RES	6, A		F7	SET	6, A	
38	SRL	B		78	BIT	7, B		B8	RES	7, B		F8	SET	7, B	
39	SRL	C		79	BIT	7, C		B9	RES	7, C		F9	SET	7, C	
3A	SRL	D		7A	BIT	7, D		BA	RES	7, D		FA	SET	7, D	
3B	SRL	E		7B	BIT	7, E		BB	RES	7, E		FB	SET	7, E	
3C	SRL	H		7C	BIT	7, H		BC	RES	7, H		FC	SET	7, H	
3D	SRL	L		7D	BIT	7, L		BD	RES	7, L		FD	SET	7, L	
3E	SRL	(HL)		7E	BIT	7, (HL)		BE	RES	7, (HL)		FE	SET	7, (HL)	
3F	SRL	A		7F	BIT	7, A		BF	RES	7, A		FF	SET	7, A	

Appendix 4

DD XX			
00 01 02 03 04 05 06 07 08 09 ADD IX, BC 0A 0B 0C 0D 0E 0F	40 41 42 43 44 45 46 LD B, (IX+d) 47 48 49 4A 4B 4C 4D 4E LD C, (IX+d) 4F	80 81 82 83 84 85 86 ADD A, (IX+d) 87 88 89 8A 8B 8C 8D 8E ADC A, (IX+d) 8F	C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB  CC CD CE CF
10 11 12 13 14 15 16 17 18 19 ADD IX, DE 1A 1B 1C 1D 1E 1F	50 51 52 53 54 55 56 LD D, (IX+d) 57 58 59 5A 5B 5C 5D 5E LD E, (IX+d) 5F	90 91 92 93 94 95 96 SUB (IX+d) 97 98 99 9A 9B 9C 9D 9E SBC A, (IX+d) 9F	D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
20 21 LD IX, nn 22 LD (nn), IX 23 INC IX 24 25 26 27 28 29 ADD IX, IX 2A LD IX, (nn) 2B DEC IX 2C 2D 2E 2F	60 61 62 63 64 65 66 LD H, (IX+d) 67 68 69 6A 6B 6C 6D 6E LD L, (IX+d) 6F	A0 A1 A2 A3 A4 A5 A6 AND (IX+d) A7 A8 A9 AA AB AC AD AE XOR (IX+d) AF	E0 E1 POP IX E2 E3 EX (SP), IX E4 E5 PUSH IX E6 E7 E8 E9 JP (IX) EA EB EC ED EE EF
30 31 32 33 34 INC (IX+d) 35 DEC (IX+d) 36 LD (IX+d), n 37 38 39 ADD IX, SP 3A 3B 3C 3D 3E 3F	70 LD (IX+d), B 71 LD (IX+d), C 72 LD (IX+d), D 73 LD (IX+d), E 74 LD (IX+d), H 75 LD (IX+d), L 76 77 LD (IX+d), A 78 79 7A 7B 7C 7D 7E LD A, (IX+d) 7F	B0 B1 B2 B3 B4 B5 B6 OR (IX+d) B7 B8 B9 BA BB BC BD BE CP (IX+d) BF	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 LD SP, IX FA FB FC FD FE FF

DD CB -d- XX			
00 01 02 03 04 05 06 RLC (IX+d) 07 08 09 0A 0B 0C 0D 0E RRC (IX+d) 0F	40 41 42 43 44 45 46 BIT 0, (IX+d) 47 48 49 4A 4B 4C 4D 4E BIT 1, (IX+d) 4F	80 81 82 83 84 85 86 RES 0, (IX+d) 87 88 89 8A 8B 8C 8D 8E RES 1, (IX+d) 8F	C0 C1 C2 C3 C4 C5 C6 SET 0, (IX+d) C7 C8 C9 CA CB CC CD CE SET 1, (IX+d) CF
10 11 12 13 14 15 16 RL (IX+d) 17 18 19 1A 1B 1C 1D 1E RR (IX+d) 1F	50 51 52 53 54 55 56 BIT 2, (IX+d) 57 58 59 5A 5B 5C 5D 5E BIT 3, (IX+d) 5F	90 91 92 93 94 95 96 RES 2, (IX+d) 97 98 99 9A 9B 9C 9D 9E RES 3, (IX+d) 9F	D0 D1 D2 D3 D4 D5 D6 SET 2, (IX+d) D7 D8 D9 DA DB DC DD DE SET 3, (IX+d) DF
20 21 22 23 24 25 26 SLA (IX+d) 27 28 29 2A 2B 2C 2D 2E SRA (IX+d) 2F	60 61 62 63 64 65 66 BIT 4, (IX+d) 67 68 69 6A 6B 6C 6D 6E BIT 5, (IX+d) 6F	A0 A1 A2 A3 A4 A5 A6 RES 4, (IX+d) A7 A8 A9 AA AB AC AD AE RES 5, (IX+d) AF	E0 E1 E2 E3 E4 E5 E6 SET 4, (IX+d) E7 E8 E9 EA EB EC ED EE SET 5, (IX+d) EF
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E SRL (IX+d) 3F	70 71 72 73 74 75 76 BIT 6, (IX+d) 77 78 79 7A 7B 7C 7D 7E BIT 7, (IX+d) 7F	B0 B1 B2 B3 B4 B5 B6 RES 6, (IX+d) B7 B8 B9 BA BB BC BD BE RES 7, (IX+d) BF	F0 F1 F2 F3 F4 F5 F6 SET 6, (IX+d) F7 F8 F9 FA FB FC FD FE SET 7, (IX+d) FF

Appendix 4

ED XX			
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	40 IN B, (C) 41 OUT (C), B 42 SBC HL, BC 43 LD (nn), BC 44 NEG 45 RETN 46 IM 0 47 LD I, A 48 IN C, (C) 49 OUT (C), C 4A ADC HL, BC 4B LD BC, (nn) 4C 4D RETI 4E 4F LD R, A	80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F	C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F	50 IN D, (C) 51 OUT (C), D 52 SBC HL, DE 53 LD (nn), DE 54 55 56 IM 1 57 LD A, I 58 IN E, (C) 59 OUT (C), E 5A ADC HL, DE 5B LD DE, (nn) 5C 5D 5E IM 2 5F LD A, R	90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F	D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F	60 IN H, (C) 61 OUT (C), H 62 SBC HL, HL 63 64 65 66 67 RRD 68 IN L, (C) 69 OUT (C), L 6A ADC HL, HL 6B 6C 6D 6E 6F RLD	A0 LDI A1 CPI A2 INI A3 OUTI A4 A5 A6 A7 A8 LDD A9 CPD AA IND AB OUTD AC AD AE AF	E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F	70 71 72 SBC HL, SP 73 LD (nn), SP 74 75 76 77 78 IN A, (C) 79 OUT (C), A 7A ADC HL, SP 7B LD SP, (nn) 7C 7D 7E 7F	B0 LDIR B1 CPIR B2 INIR B3 OTIR B4 B5 B6 B7 B8 LDDR B9 CPDR BA INDR BB OTDR BC BD BE BF	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

FD XX			
00 01 02 03 04 05 06 07 08 09 ADD IY, BC 0A 0B 0C 0D 0E 0F	40 41 42 43 44 45 46 LD B, (IY+d) 47 48 49 4A 4B 4C 4D 4E LD C, (IY+d) 4F	80 81 82 83 84 85 86 ADD A, (IY+d) 87 88 89 8A 8B 8C 8D 8E ADC A, (IY+d) 8F	C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB <input type="text"/> CC CD CE CF
10 11 12 13 14 15 16 17 18 19 ADD IY, DE 1A 1B 1C 1D 1E 1F	50 51 52 53 54 55 56 LD D, (IY+d) 57 58 59 5A 5B 5C 5D 5E LD E, (IY+d) 5F	90 91 92 93 94 95 96 SUB (IY+d) 97 98 99 9A 9B 9C 9D 9E SBC A, (IY+d) 9F	D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
20 21 LD IY, nn 22 LD (nn), IY 23 INC IY 24 25 26 27 28 29 ADD IY, IY 2A LD IY, (nn) 2B DEC IY 2C 2D 2E 2F	60 61 62 63 64 65 66 LD H, (IY+d) 67 68 69 6A 6B 6C 6D 6E LD L, (IY+d) 6F	A0 A1 A2 A3 A4 A5 A6 AND (IY+d) A7 A8 A9 AA AB AC AD AE XOR (IY+d) AF	E0 E1 POP IY E2 E3 EX (SP), IY E4 E5 PUSH IY E6 E7 E8 E9 JP (IY) EA EB EC ED EE EF
30 31 32 33 34 INC (IY+d) 35 DEC (IY+d) 36 LD (IY+d), n 37 38 39 ADD IY, SP 3A 3B 3C 3D 3E 3F	70 LD (IY+d), B 71 LD (IY+d), C 72 LD (IY+d), D 73 LD (IY+d), E 74 LD (IY+d), H 75 LD (IY+d), L 76 77 LD (IY+d), A 78 79 7A 7B 7C 7D 7E LD A, (IY+d) 7F	B0 B1 B2 B3 B4 B5 B6 OR (IY+d) B7 B8 B9 BA BB BC BD BE CP (IY+d) BF	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 LD SP, IY FA FB FC FD FE FF

Appendix 4

FD CB -d- XX			
00 01 02 03 04 05 06 RLC (IY+d) 07 08 09 0A 0B 0C 0D 0E RRC (IY+d) 0F	40 41 42 43 44 45 46 BIT 0, (IY+d) 47 48 49 4A 4B 4C 4D 4E BIT 1, (IY+d) 4F	80 81 82 83 84 85 86 RES 0, (IY+d) 87 88 89 8A 8B 8C 8D 8E RES 1, (IY+d) 8F	C0 C1 C2 C3 C4 C5 C6 SET 0, (IY+d) C7 C8 C9 CA CB CC CD CE SET 1, (IY+d) CF
10 11 12 13 14 15 16 RL (IY+d) 17 18 19 1A 1B 1C 1D 1E RR (IY+d) 1F	50 51 52 53 54 55 56 BIT 2, (IY+d) 57 58 59 5A 5B 5C 5D 5E BIT 3, (IY+d) 5F	90 91 92 93 94 95 96 RES 2, (IY+d) 97 98 99 9A 9B 9C 9D 9E RES 3, (IY+d) 9F	D0 D1 D2 D3 D4 D5 D6 SET 2, (IY+d) D7 D8 D9 DA DB DC DD DE SET 3, (IY+d) DF
20 21 22 23 24 25 26 SLA (IY+d) 27 28 29 2A 2B 2C 2D 2E SRA (IY+d) 2F	60 61 62 63 64 65 66 BIT 4, (IY+d) 67 68 69 6A 6B 6C 6D 6E BIT 5, (IY+d) 6F	A0 A1 A2 A3 A4 A5 A6 RES 4, (IY+d) A7 A8 A9 AA AB AC AD AE RES 5, (IY+d) AF	E0 E1 E2 E3 E4 E5 E6 SET 4, (IY+d) E7 E8 E9 EA EB EC ED EE SET 5, (IY+d) EF
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E SRL (IY+d) 3F	70 71 72 73 74 75 76 BIT 6, (IY+d) 77 78 79 7A 7B 7C 7D 7E BIT 7, (IY+d) 7F	B0 B1 B2 B3 B4 B5 B6 RES 6, (IY+d) B7 B8 B9 BA BB BC BD BE RES 7, (IY+d) BF	F0 F1 F2 F3 F4 F5 F6 SET 6, (IY+d) F7 F8 F9 FA FB FC FD FE SET 7, (IY+d) FF

索引

A

アクセス17
 ・タイム15, 17
アキュムレータ35
アドレス14
 ・バス16
 I/O16
アドレッシング〈・モード〉40
 ビット41
 エクステンド40
 インデックス41
 インプライ41
 イミディエイト40
 イミディエイト・エクステンド40
 モディファイ・ページ・ゼロ41
 レジスタ40
 レジスタ・インダイレクト40
 レラティブ41
 絶対40
アルゴリズム16
オルタネート46
ALU13
アナログ注34
AND(論理積)29, 注48
ASCIIコード76, 注98
アセンブル6
 ・リスト注90
 ハンド6
アセンブラ7, 74
 言語39
 モニタ72
 セルフ80

B

BASIC2
 MSX2, 注2
バンク切り換え注87
BCD23
ベン図表29
バイナリ(2進数)5, 18, 19
ビット14
 ・アドレッシング41
 操作39, 60
 符号22
バイト14
ブール代数29
ボロー〈フラグ〉(借り)27, 28, 注46
暴走4, 注6
ブレーク・ポイント78
ブロック
 ・サーチ39, 47, 49
 転送39, 47
バス13, 注23
 アドレス16
 コントロール17
 データ16

C

コール39, 62
キャリー〈フラグ〉(桁上がり)25, 26, 35, 注44
 ハーフ36
チェック・サム77, 87
クロック34, 注31
COBOL12, 注20
コード
 ASCII76, 注98
 マシン39
コーディング115
コマンド74
コンパイラ4
コントロール・バス17
カウンタ
 プログラム36
 ロケーション82, 84
CPU(中央処理装置)5, 13, 注15
 コントロール39, 69
カーソル(スクリーン)・エディット76
サイクル34
 ・タイム15

D

ダンプ
 メモリ76
データ・バス16
デバッグ72, 114, 注93
デクリメント55
デスティネーション42
ディテール・フローチャート115
ダイナミックRAM37, 注65
デジタル注34
ディスプレイ15
ディスプレイメント41
ドット注35

E

エディット(エディタ)注86, 注94
 カーソル(スクリーン)76
演算12, 13, 25, 39, 51
 算術25
 論理25, 29
EQV(同値)29
エラー86
 リスト74
エクステンド・アドレッシング40
 イミディエイト40

F

FORTRAN12, 注20
フラグ14
 操作39, 60
 ・レジスタ35
キャリー35
ボロー28
減算35

パリティ/オーバーフロー	36
ハーフ・キャリー	36
ゼロ	36
サイン	36
フローチャート(流れ図)	114, 116
ディテール	115
ゼネラル	注105

G

ゼネラル・フローチャート	注105
減算フラグ	35
擬似命令	82
偽	29
グラフィック	注19
行番号	3, 注3

H

ハーフ・キャリー(フラグ)	36
ハンド・アセンブル	6
汎用レジスタ	36, 注24
ハードウェア	12
補数	
1の	注38
2の	22
浮動小数点	23
符号ビット	22
負数	21

I

IFF(割込み許可フリップ・フロップ)	注82
IMP(包含)	29
イミディエイト	
・アドレッシング	40
・エクステンド・アドレッシング	40
インクリメント	36, 54
インデックス	37
・アドレッシング	41
・修飾	41
イニシャライズ(初期設定)	73
インブライ・アドレッシング	41
インストラクション・デコーダ&CPUコントロール	13
インターフェイス	15
インタープリタ	4, 注10, 注86
インタラプト(割込み)	37, 38
インターバル・タイマ	38
IO	
アドレス	16
インターフェイス	15
ポート	16

J

ジャンプ	39, 62
相対	63

K

仮数	24
検索	47
桁上がり	25

記号	6
記憶	12
・装置	13
小型計算機	13, 注22
交換	39, 41
高級言語	注7

L

ラベル	81
・リスト	74
リスト	
アセンブル	注90
プログラム	74
エラー	74
ロード	39, 41, 42
ロケーション・カウンタ	82, 84

M

マシン	
・コード	39
・サイクル	34
語	注6
メイン・ルーチン	注80
マスカブル	38
メモリ	3, 14, 注4
・ダンブ	76
・マップ	15, 注88
マイクロ・プロセッサ	34
モード	
アドレッシング	40
テキスト	72
割込み	38
モディファイ・ページ・ゼロ・アドレッシング	41
文字列	75
モニタ	5
・アセンブラ	72
MSX-BASIC	2, 注2

N

流れ図	114, 116
記号	116
ニモニク	6, 39, 82
入力	12
入出力装置	13, 15
ノンマスカブル	38
NOT(否定)	29, 注48
ノウハウ	5

O

オブジェクト	72, 注21
オフセット	79
オペコード	34
オペランド	39, 83
ON(OFF)	14
OR(論理和)	29, 38, 注48
オーバーフロー	
パリティ	36

P

ペア・レジスタ	36
パラメータ(引数)	115
パリティ/オーバーフロー<フラグ>	36
パス	80
パターン	19
ポイント	
ブレーク・ー	78
ポート	16
プリンタ	15, 注28
プログラム	2
ー・カウンタ	36
ー・リスト	74
ソース・	72, 80

R

ラディックス(基数)	注32
RAM	14
ダイナミック・ー	37, 注65
スタティック・ー	注65
V・ー	72
リード	14
リフレッシュ	37, 注65
レジスタ	5, 13, 注18
ー・アドレッシング	40
ー・インダイレクト・アドレッシング	40
フラグ・ー	35, 注25
汎用・ー	36, 注24
ペア・ー	36
裏・ー	注73
レラティブ・アドレッシング	41
リセット	30, 60, 注31
ーボタン	注83
リターン	39, 62
ROM	14
ーOS	117
論理	
ー演算	25, 29
ーシフト	31
ローテイト(回転)	25, 31, 39, 58
ルーチン	
メイン・ー	注80

S

算術	
ー演算	25
ーシフト	31
スクリーン	72
サーチ	47
ブロック・ー	39, 47, 49
ストリング・ー	72, 75
制御	12, 13, 注3
正規化	注42
セルフ・アセンブラ	80
セット	30, 60
シフト(移動)	25, 31, 39, 58
小数	21, 注9

出力	12
サイン<フラグ>	36
真	29
信号線	13
真理値表	29
指数	24
仕様	114, 117
ソフトウェア	12
ソース	42
・プログラム	72, 80
相対ジャンプ	63
スプライト(動画)	2, 注1
スタック<ポインタ>	37, 42, 注62
ステート	34
数	35
スタティックRAM	注65
ストリング・サーチ	72, 75
サブルーチン	注80
数	18
数字	18

T

定数	83
転送	39, 41
テスト	60
テキスト・モード	72

U

裏レジスタ	注73
USR関数	3, 118, 注5

V

VDP	38, 72
VRAM	72

W

WAIT	34
割込み	37, 38, 注31
モード	38
ライト	14

X

XOR(排他的論理和)	29, 注49
-------------	---------

Z

Z80A	5, 34, 注17
ゼロ<フラグ>	36
絶対アドレッシング	40

数字

1の補数	注38
2の補数	22
2進法	19
2進数(バイナリ)	5, 18, 19
10進法	19
10進数	18, 19
16進法	20
16進数	5, 18, 20

MSX FANシリーズ1 **マシン語入門(基礎編)**

1984年4月1日 初版発行
1985年5月1日 2刷発行

著 者 大貫広幸
発行人 塚本慶一郎
発行所 株式会社 エム・アイ・エー
〒150 東京都渋谷区渋谷2-9-1 青山田中ビル
電話(03)486-4500(代)
イラスト 関 寿泰 大村敦郎
印刷・製本 有限会社 チトセ印刷

ISBN 4-87170-007-0 C3055

||||| 料金受
取人払 ||||| 郵便はがき

渋谷局
承認
2444

差出有効期間
昭和60年12月
31日まで

150-□□

東京都渋谷区渋谷二一九一
青山田中ビル
(株)エム・アイ・エー
マシン語入門(基礎編)
編集部 係

フリガナ			性別	男・女
氏名			年令	才
ご住所	〒 Tel.()			
勤務先・所属部課 学校名・学科名				
お持ちのマイコン		使用する プログラム言語		
マイコン・クラブ			経験	年

アンケート

本誌をお買いあげいただきありがとうございました。みなさんのご意見を編集の参考にさせていただきたいと思います。アンケートにご協力ください。

① 本誌を何でお知りになりましたか。

- 書店〔 〕 ●広告〔 〕 ●友人、知人〔 〕
- パソコンショップ〔 〕 ●その他〔 〕
- 店名〔 〕

② 本誌をどこでお買いになりましたか。

- 書店〔 〕 ●パソコンショップ〔 〕
- その他〔 〕
- 店名〔 〕

③ 本誌の内容はいかがでしたか。感想をお書きください。

④ 本誌内容の編集部へのご意見・ご希望をお聞かせください。

⑤ 今後とりあげてほしい企画がありましたらお書きください。

MIA COMPUTER BOOKS

MSX *fan series*

MACHINE LANGUAGE

MIA
MICRO INFORMATION ASSOCIATES

ISBN4-87170-007-0 C3055 ¥1800E

定価1800円